SPARTAN II: AN INSTRUCTIONAL HIGH RESOLUTION LAND COMBAT MODEL

THESIS

Edwin H. Harris III, Captain, USA
AFIT/GOR/ENS/93M-09

AFIT/GOR/ENS/93M-09

SPARTAN II: AN INSTRUCTIONAL HIGH RESOLUTION LAND COMBAT MODEL

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In partial fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Edwin H. Harris III, B.S.

Captain, USA

March 1993

THESIS APPROVAL

STUDENT: CPT Edwin Hawkins Harris III CLASS: GOR-93M

THESIS TITLE: SPARTAN II: AN INSTRUCTIONAL HIGH RESOLUTION LAND COMBAT MODEL

DEFENSE DATE: 24 February 1993

COMMITTEE	NAME/DEPARTMENT	SIGNATURE
ADVISOR	MAJ Edward Negrelli/ENS	[illegible]
READER	MAJ Bruce Morlan/MA	[illegible]

Preface

The goal of this thesis was to improve SPARTAN, a high resolution land combat model developed for use in a land combat modeling course. Using the existing model concept, the subsequent model development progressed through code development and initial implementation. The new SPARTAN models more combat processes and more accurately displays current modeling techniques than the previous version did. The model also is simple enough for students to readily understand model components and their operation.

This thesis provides background information about the original SPARTAN, outlines the development methodology, and discusses the methods employed to model combat processes. It also provides computational templates, a user's manual, and the computer code. As designed, SPARTAN is an improved model (although still a simple one), that should serve as a useful tool for learning about the advantages and disadvantages of high resolution combat modeling.

I wish to thank MAJ Edward Negrelli and MAJ Bruce Morlan for their guidance and invaluable assistance in the development of this thesis. I also wish to thank the other officers in the land combat modeling courses who proudly served as guinea pigs and provided outstanding feedback about model shortcomings. Finally, I wish to thank my wife for tolerating the computer hermit for the past 8 months.

Edwin H. Harris

Table of Contents

List of Figures

List of Tables

AFIT/GOR/ENS/93M-09

Abstract

This project improved SPARTAN, a high resolution land combat model demonstrator. SPARTAN was originally developed as a hands-on trainer for land combat modeling students because there were no models specifically designed to teach the analysts how the models function.

SPARTAN is built to demonstrate the techniques used in the current generation of US Army high resolution models. Like the original, the model is primarily a small scale attrition (both direct and indirect fire) model. The model represents 12 soldiers involved in the following processes: target search, target selection, direct fire engagement, indirect fire engagement, movement, reaction to fire, obstacle breaching, and some elements of command and control. The emphasis on model development was to keep the logic simple, yet accurately portray current high resolution modeling techniques as used in the more capable models, JANUS and the Combined Arms and Support Task force Evaluation Model (CASTFOREM). SPARTAN contains numerous features that allow the user to observe, in great detail, how the model represents the various activities of the soldiers. An educational assessment of the model was performed by students and faculty at the Air Force Institute of Technology.

SPARTAN II: AN INSTRUCTIONAL HIGH RESOLUTION LAND COMBAT MODEL

I. Introduction

General

The purpose of this thesis is to document my improvements tc SPARTAN, the prototype high resolution land combat model demonstrator. This chapter provides background information about my efforts to improve SPARTAN's value as an instructional tool. The first section of this chapter reviews the increasing importance of simulations to the Army and some of the problems with simulation use. The next section identifies the problem that SPARTAN attempts to solve and the specific objectives of my improvements to SPARTAN. The next section defines some important modeling terms that are used throughout this thesis. This is followed by an outline of the scope of my improvements to SPARTAN and the approach that I employed in making these improvements. I then identify the equipment requirements for SPARTAN. The last section will outline the thesis organization.

Background

A model is "an idealization, ar abstraction of a part of the real world" (20:211) and can be a set of mathematical equations, a scenario, a board game cr a computer simulation. The military uses models and simulations to accomplish a variety of tasks such as (4:6):

1) Technical evaluation
2) Doctrinal evaluation
3) Force-structure evaluation
4) Analysis of military and diplomatic factors and international relations
5) Training and education
6) Development of research methodology

Thus, one military model might estimate the best way to allocate nuclear bombers and missiles to ensure destruction of enemy targets while another might evaluate the effectiveness of new infantry rifles.

The Army has been interested in models for all of the reasons above, but has been particularly interested in models for force structure and doctrinal evaluation, and for training and education. Over the years, the Army has developed models that simulate theater level conflicts and other models and simulations that replicate single vehicles and soldiers. These models are used at every echelon of the Army, from the vehicle crew level to Corps and Army staffs and commanders. One primary reason for this increased use of models and simulations is money. It costs less to practice on a machine or computer, than it does to use tanks (or other expensive equipment), personnel, and other valuable training resources. This increasing reliance on simulations shows no sign of slowing and, in fact, will probably increase as defense budgets continue to decline.

General Frederick M. Franks, Commanding General, U.S. Army Training & Doctrine Command, addressed the use and importance of simulations to the Army in an interview published in the "ARMED FORCES JOURNAL":

> We put all that together and decided that the Army needs to provide itself an institutionalized way to continue to examine if we're going in the right direction for the circumstances within which we are now called to serve the nation.
> ....the Louisiana Maneuvers. They will give us the opportunity to use simulations, which have advanced considerably in their ability to replicate the battlefield with great fidelity at all levels.....on perhaps an annual basis to test, in an operational scenario, the Army's ability to fulfill its Title 10 responsibilities...(21:67-68)

In fact, General Franks has proposed a system of "battle laboratories" where simulations will be tied in with actual field units to test and develop mcre effective tactics and training scenarios (21:67-68). Thus, simulations and models will play an increasingly important role in the Army.

While models and simulations are of great benefit to the Army, they also present a danger. This danger is that model results might be accepted as truth as opposed to being only one sample of many possible truths. Decision makers and analysts should keep this aspect of models in mind. A 1980 GAO report highlights this fact, stating that models "are intended to be used as an extension of, rather than a replacement for, human judgment" (12:9).

This danger is exacerbated in the combat models used by the Army, because these models replicate actions and decisions of humans under stress. This makes the problem of modeling combat "squishy". Squishy problems "do not have a formulation that is both analytically tractable (based on science, empirical research, etc.), and which unambiguously captures the substantive problem" (12:9). This fact makes modeling such problems difficult and the formulation and conclusions for them are "inherently subjective--requiring and depending on careful and considered judgment by the decision maker" (12:9).

Thus, the increased reliance of the Army on simulations and models is going to require an increased understanding on the part of decision makers about the way combat processes are modeled. The problem is that combat simulations tend to be complex, modeling thousands of entities and hundreds of systems and processes. CASTFOREM (one of the Army's land combat models), for example, models combined arms ground conflicts and includes such systems as helicopters, fixed wing aircraft, air defense, and dismounted soldiers. It also takes into account weather, ambient light conditions, and battlefield obscurants. It models direct fire weapons, directed energy weapons, and indirect fire weapons (14:C-1;6:1-11). The complexity caused by modeling all of this ensures that the decision maker will probably not know what assumptions and modeling techniques the model incorporates. Therefore, he probably does not understand the quality of the data on which he is basing his decision. Thus, it is extremely important that the analyst/modeler performing the study or analysis understands the assumptions and methods used in the model and is able to communicate the advantages and disadvantages of these to the decision maker.

Unfortunately there is very little written about modeling combat processes and there are no simulations or models designed specifically to train analysts about modeling these processes. In an effort to correct this problem, CPT David Cox, a 1992 Air Force Institute of Technology Operations Research masters student, developed a prototype land combat model demonstrator called SPARTAN. SPARTAN is a high resolution combat model "developed for use as an instructional aid in land combat modeling courses" (5:ii). It uses modeling techniques that are representative of the techniques used in the two premier Army high resolution models, CASTFOREM and JANUS (5:ii).

Problem Statement

The Army makes extensive use of high resolution land combat models for analysis and to train brigade and battalion commanders and staffs. A search of the 528 games, models and simulations listed in the Catalogue of Wargaming and Military Simulations Modeling however, reveals that there is no suitable model to train the analysts who use these models (14:iii). CPT David Cox (GOR-M92) developed SPARTAN to fill this need. SPARTAN does demonstrate some of the basic current modeling techniques, but leaves much room for improvement.

The purpose of this thesis effort is to continue development of SPARTAN, and improve its value as an instructional tool for land combat modeling courses.

Objectives

SPARTAN is a war game. A wargame can be defined as:

> a simulated military operation involving two or more opposing forces and using rules, data, and procedures designed to depict an actual or hypothetical real-life situation. It is used primarily to study problems of military planning, organization, tactics, and strategy. (4:8)

Because of the broad range of modeling activities under the general heading of war games, there are few hard rules for designing them. There are, however, "certain principles of modeling which are international and, in fact, transcend technique, military focus, or specific model" (2:9). Because these are principles and not rules, building a war game/combat model becomes something of an art. The successful combat modeler should ensure the model (4:9-11):

1) Fits customer requirements.
2) Produces the appropriate criteria for analysis.
3) Resolves the conflict between the desired level of modeling detail and uncertainties in input data.
4) Deals with uncertainties either explicitly, implicitly, parametrically or by hedging.

5) Is in great breadth or in great detail, but not both.

6) Becomes more rigid as the focus of the application becomes closer to chronological time.

7) Does not exceed the capabilities and level of detail as constrained by the state of the art of the computing equipment.

8) Is invisible to the user.

Using these principles, I decided upon the following objectives in continuing the development of SPARTAN: portability, simplicity, applicability, and usability.

By portability, I refer to the ease of transferring SPARTAN. This means that SPARTAN (and its supporting data files) is small enough (less than 500K) to be kept on a single disc and is written in a generally available and easily decipherable language (QuickBASIC). It also means that SPARTAN does not require a super computer. It is, in fact, designed to be run on a personal computer.

Simplicity refers to the number of combat processes SPARTAN demonstrates. This number is kept to a minimum to avoid complexity. Complexity would defeat SPARTAN's purpose of being an easily understood instructional tool and would also limit its portability.

SPARTAN is also applicable. The processes, as far as possible, accurately portray techniques used by JANUS and CASTFOREM. These techniques may or may not be good replications of reality, but by using SPARTAN, students will gain insight into the implications of using these techniques.

SPARTAN is usable. It produces output, again similar to the output from CASTFOREM and JANUS, that can be used for analysis. SPARTAN accepts varied input so sensitivity analysis can be performed. SPARTAN also is designed so that users need not be infantry experts or computer geniuses to run the simulation.

Definitions

This section defines a few modeling terms that will be used throughout the remainder of this thesis.

High Resolution. A high resolution combat model is a model that simulates activities of individual entities, rather than aggregated units. An entity is an individual soldier, tank, cannon, or aircraft etc. as opposed to an aggregate (unit) composed of two or more individuals. Each individual entity has its own characteristics and the interactions of these individuals determine the outcome of conflicts. The individual combatant sees the battle from his own perspective and conflicts are resolved on an individual shooter-target basis. The emphasis on detail makes representation believable, but limits the number of combatants that can be modeled. This is because the thousands of entities doing several processes overwhelm the ability of the computer (13:1-6,1-7).

Stochastic Process. Most high resolution combat models use stochastic processes to model uncertainty. A stochastic process is a process where the next event is decided in a probabilistic manner. There may be several possible next events, some more likely to occur than others. This type of process is particularly appropriate when describing combat because of the uncertainty of outcomes in battle. Using models that employ stochastic processes to determine outcomes, injects uncertainty and chance outcomes and provides a more realistic representation of combat (13:1-6). Models frequently employ the Monte Carlo method to achieve this probabilistic outcome. Using this method, a random number is drawn from a distribution and compared to a threshold probability of some event occurring. If the random number is less than the threshold, the event occurs, otherwise, the event does not occur.

<u>Interactive Model</u>. All models are, in reality, interactive in that they require some form of user input. This can be either data input prior to running the model or decision input during model runs (or a combination of both). I will refer to interactive models as ones in which users can provide input (interact with the computer) during model runs to influence model outcomes. In combat models, these inputs take the form of tactical decisions such as where to move units, who to engage with indirect or direct fire, who to supply, and other realistic battlefield decisions. This type of model is very useful as a training tool because it allows people to react to situations, but is less useful for analysis because the input is not controlled. This makes it

difficult to determine an audit trail of causal relationships.

Scenario. The scenario is the situation that establishes the initial conditions for the model. The scenario includes friendly and enemy force structures, weapons, plans, and tactics. Scenarios also include terrain and environmental considerations.

Terrain representation. Terrain representation is one of the most complicated and most important aspects of land combat models. Normally, terrain is represented by subdividing it into regularly shaped polygons of various sizes. Each of these polygons normally contain terrain of the same characteristics, such as slope, vegetation, and trafficability. These characteristics are assigned values and become the polygon's attributes.

Movement. Land combat models generally must simulate the movement of combatants across the battlefield. Rates of advance depend on attributes of the moving entity such as speed, weight, cross country mobility, and formation and on the attributes of the terrain polygon through which the entity is moving.

Target Engagement. Target engagement is actually composed of the three sequential subprocesses of target detection, target selection and target engagement (firing the weapon). All three of these processes are stochastic. Entities search an area and have a probability of detecting a target. If more than one target is detected, the entity then rank orders his detected targets based on target prioritization rules given by the model or by the user. Based on these rules he may or may not fire. If he fires, he has a probability of success of hitting the target based on range, size of target and weapon accuracy. If he hits the target, he has a probability of either wounding or killing the target.

<u>Scope</u>

CPT Cox was successful in designing a prototype model that demonstrated basic high resolution combat modeling techniques. CPT Cox's version of SPARTAN demonstrated the following (5:94):

1) Time keeping and an implementation technique for event set management and synchronization.

2) Algorithms used to model movement, terrain, target detection, target selection, weapon accuracy, and attrition.

3) Techniques to model decision logic of the soldier as well as simple command and control issues.

4) Stochastic techniques for representing the occurrence of randomness on the battlefield.

5) Data requirements and storage techniques for various model components.

6) The overall process of developing a combat simulation model from concept to implementation.

7) An example of components for a typical combat model such as the scenario input, a preprocessor, the simulation model, various types of output, and accompanying documentation.

There were, however, several areas I felt could be improved. These areas are (5:94-100):

1) Computer graphics. They were simplistic and did not enhance understanding of model processes.

2) Movement processes. Movement is not tied to any formation. All entities move independently.

3) Terrain issues. There is no obstacle play and terrain detail is lacking.

4) Weapons. CPT Cox used made up weapons accuracy data and made up weapons. There is no indirect fire modeling.

6) Preprocessor activities. Using the preprocessor is difficult. It does not make clear what the effect of altering data files will have on the simulation.

I limited the scope of my improvements to these areas.

Approach

There are ten stages of development of simulation models. These are (19:10-11):

1) Problem Formulation

2) Model Building

3) Data Acquisition

4) Model Translation

5) Verification

6) Validation

7) Strategic and Tactical Planning

8) Experimentation

9) Analysis of results

10) Implementation and documentation

This chapter serves to accomplish the first stage of this process, problem formulation. Given the nature of this problem and the purpose of SPARTAN, I focused my thesis efforts on stages 2-6 to make SPARTAN a more effective high resolution combat model demonstrator.

CPT Cox had already done stage 2 to some degree, but because of the improvements and additions that I made to SPARTAN, it was necessary to redesign his model. Only a few of the original model components remain.

The acquisition of data necessarily supported my improvements of SPARTAN. I continued to rely on James Hartman's unpublished notes on high resolution combat modeling, as did Cox, for model methodology and theory. Army Field Manuals were helpful for realistic force structuring, movement formations, and movement speeds. This in turn helped demonstrate model invisibility. I received weapons accuracy data from the Army Material Systems Analysis Activity (AMSAA) for both US and Soviet squad weapons, this data combined with Engineering Design Handbook, Army Weapon Systems Analysis, Part One, helped in modeling different weapons' effects. Since the purpose of SPARTAN is to demonstrate techniques used in the Army's two premier models, I compared all of SPARTAN's combat processes to those modeled in CASTFOREM and JANUS. Another major improvement was the modeling of indirect fire. The CASTFOREM and JANUS manuals and FM 23-91 Mortar Gunnery were helpful for this.

The big step in making improvements was the translation of improvements into computer code. CPT Cox tried, as much as possible, to use structured block programming to make

SPARTAN. I also made use of structured programming because it is easier to troubleshoot. QuickBASIC supported this approach. I did the most difficult tasks first, such as improving graphics, interface and movement, since these carried over into all other aspects of the model.

Verification is the process of ensuring that model components are functioning as intended (19:11). This was an ongoing process. As I made programming changes and additions, I verified they did what I wanted them to. I continuously documented my efforts so that follow on work could be done.

The last stage of the model development process is that of validation. Validation is the process of ensuring that model results are correct (19:11). This was difficult as there are "no experimentally verified models of combat processes" (3:76). Instead of validation, I used the same validation criteria as CPT Cox, that of reasonableness (face validity). Since SPARTAN's results looked reasonable, I accepted them as valid. I also used blind testing, issuing documentation and the program to AFIT land combat modeling students to see if SPARTAN achieved its intended purpose, that of accurately demonstrating high resolution land combat modeling techniques.

Equipment

The equipment that I used for this thesis is the same as listed by Cox:

1) IBM AT compatible microcomputer with minimum EGA color graphics

2) Microsoft DOS 3.3

3) Microsoft QuickBASIC 4.5 programming language

4) MATHCAD 2.5 Mathematics Software

SPARTAN runs on any IBM or IBM clone with DOS 3.0 or better. I designed and built SPARTAN on a 286 and have run it on 486s. Obviously, it runs faster on 486, but performance was satisfactory on the 286 models.

Thesis Organization

Chapter II is a review of specific SPARTAN combat processes and a comparison of them to processes modeled in JANUS and CASTFOREM. Chapter III is a review of the methodology I employed in formulating, building, and coding SPARTAN. Chapter IV covers the specific algorithms and logic I employed in SPARTAN's combat processes. Chapter V summarizes my efforts and offers recommendations for future SPARTAN improvements. Appendix A is a MATHCAD 2.5 template used for computing threshold observer-target probabilities. Appendix B is a MATHCAD 2.5 template used for computing the probability of acquisition tables. Appendix C is a MATHCAD 2.5 template for computing the probability of hit for the M16A2. It also includes raw accuracy data for all weapons used in SPARTAN and the appropriate probability of hit tables. Appendix D is the user's guide. Appendix E is a listing of computer code for the preprocessor. Appendix F is a listing of computer code for the simulation.

II. SPARTAN and Combat Modeling Review

Introduction

The purpose of this chapter is to review the techniques and methodology used in the original SPARTAN and compare and contrast them to techniques used in current high resolution land combat models. The first section reviews background information about SPARTAN and the two premier Army high resolution models, JANUS and CASTFOREM. The following section discusses the general composition of SPARTAN. The third section concentrates on the combat processes that SPARTAN initially covered, and the combat processes that I added. The last section reviews SPARTAN simulation output.

SPARTAN Background Information

SPARTAN is a high resolution land combat model designed to demonstrate current modeling techniques. It is "primarily a small scale direct fire attrition model under the definitions of the Army Model Improvement Program" (5:13). Using the definitions provided in the Catalog of Wargaming and Military Simulations and Models (16:A1-A22), SPARTAN is a high resolution, two-sided, force-on-force, stochastic, event sequenced simulation model of a direct fire conflict. I will go into detail what these terms mean later in this chapter.

SPARTAN was developed because of the lack of a suitable model to train analysts who would be working with high resolution combat models in the Army. CPT Dave Cox, an Army student at the Air Force Institute of Technology, built SPARTAN from the ground up in 1991-1992 at the request of MAJ Garambone, the AFIT land combat modeling instructor. With refinement, the model was for use at AFIT's Land Combat Modeling course.

Parent Models

SPARTAN was designed to demonstrate basic combat modeling procedures like those used in the Army's premier high resolution land combat models, JANUS and the Combined Arms and Support Task Force Evaluation Model (CASTFOREM) (5:95). The proponent for both of these models is the Army's Training and Doctrine Command Analysis Center-White Sands Missile Range (TRAC-WSMR), based at White Sands Missile Range, New Mexico (14: C1,J4). TRAC uses these two models for doctrinal and force-structure evaluation and for training and education at the brigade and battalion level.

TRAC developed CASTFOREM at White Sands Missile Range in 1983 and has continued to update it (14:C-11). CASTFOREM is a "high resolution, two-sided, force-on-force, stochastic, event sequenced, systemic simulation model of a combined arms conflict" (7:1-2). TRAC uses it primarily for analysis of tactics and force structure. CASTFOREM models all types of direct fire, helicopters, dismounted infantry, artillery, engineering operations, logistics, combat service support operations, communications, maneuver, detailed search and acquisition and realistic battlefield conditions (7:2-3). The model is non-interactive; once initial data has been input to the computer, there is no requirement for human decisions as inputs. The entities in the model have their own characteristics and select their own actions using decision tables.

The Conflict Simulation Laboratory at Lawrence Livermore National Laboratory developed JANUS in 1978. Lawrence Livermore and TRAC have developed several versions, four of which are still in use (14:J-3). JANUS(T) is similar to CASTFOREM in resolution and methodologies, except that it is interactive. In the interactive mode, JANUS(T) requires that human decisions be input during conflict evaluations. This mode is the one employed for training brigade and battalion staffs.

SPARTAN Hardware/Software Requirements

One of CPT Cox's original goals in developing SPARTAN "was to provide a structured program code that would be easy to understand and enhance as desired" (5:66). He looked at several languages including C+ and FORTRAN and elected to use QuickBASIC 4.5. He did this for several reasons. One reason was that QuickBASIC is relatively easy to understand. It is similar to FORTRAN in format and logic, and its editor and this simple format make it almost idiot proof. QuickBASIC also supports modular programming and has a good graphics capability. This eliminated the need for a translator or graphics program to graphically represent the simulation. QuickBASIC also has the ability to make executable files that do not require compilation in order to run.

Writing SPARTAN in QuickBASIC made it possible for CPT Cox to meet another goal, that of making SPARTAN portable (5:67). SPARTAN can be run on any IBM or IBM clone with DOS 3.3 or better. This makes it possible for users of SPARTAN to take the uncompiled version with them and make improvements or modifications as they desire. This also improves its usability because users do not have to have access to a specialized or super computer to use SPARTAN.

In addition to QuickBASIC, CPT Cox used MATHCAD 2.5 to make templates for computing probability of acquisition, probability of detection, and probability of hit tables.

SPARTAN Documentation

The documentation provided with SPARTAN served two audiences (5:63). One audience was the student who is expected to operate the model and learn about modeling. The user's manual and on-line help files were provided for this audience. The other audience was the person who wanted to get involved in SPARTAN code and modify it. CPT Cox's thesis was provided for this audience.

There was a trade-off in level of detail versus conciseness in the user's guide and the help files (5:63). CPT Cox strove to limit the size of the manual, but based on user input, he added detail to the on-line help (5:64). His user's manual is 14 pages long and provides both a general outline of model processes and explanations about why the model does what it does. The manual also explains how terrain and entities are represented. The on-line help screen provides information similar to the manual except that more detail is provided about model processes.

SPARTAN Verification and Validation

Verification is the process of ensuring that model components are functioning as intended (19:11). The modular format of SPARTAN aided CPT Cox in carrying out model verification. He was able to check processes out in detail before adding them to his model (5:55). According to Cox, "a typical improvement cycle involved creating a simple subprogram and having it print out all the data values it required" (5:55). By doing this, he was able to verify that subprograms were functioning correctly.

Validation is the process of ensuring that model results are correct (19:11). This was difficult for CPT Cox because his data was totally made up in order to provide reasonable representation of combat processes (5:95). CPT Cox did, however, institute a three level assessment process. CPT Cox evaluated the model, the thesis advisors evaluated the model, and, lastly, CPT Cox used "blind testing" (5:62). Blind testing means that the simulation and supporting documentation are given to a test user with no additional instructions (9:1-37). Testers were asked to provide feedback to CPT Cox regarding the suitability of the model for its purpose and about the documentation. Blind tests were conducted in two phases, with additions and modifications to the model and documentation occurring after each test.

SPARTAN Classification

There are several ways to classify models and simulations. One way to classify models is by purpose. Another way to classify them is by qualities of the model, such as the scope of the conflict or the level of detail of processes (1:3). A third way to classify models is by the type of construction. This includes the amount of human participation, method of time advancement and other design criteria. This section will classify SPARTAN by all of these methods.

Classification by Purpose. As was stated in Chapter I, the military uses models for (4:6):

1) Technical evaluation
2) Doctrinal evaluation
3) Force-structure evaluation
4) Analysis of military and diplomatic factors and international relations
5) Training and education
6) Development of research methodology

These categories can be divided into the two broad categories of analysis and training and education.

Models designed for analysis are typically those used for research and evaluation tools (such as engineering models) or for operations support tools (decision aids) (1:3). Typically these types of simulations are run many times to get mean expected outcomes and confidence intervals for results. Also typically, such models require little or no human interaction once the model begins a simulation run, this ensures model inputs are constant across runs. CASTFOREM and some versions of JANUS are analytic models (14:C1,J1-J4).

Models designed for training and education can be subdivided into two categories: skills development and exercise drivers (1:5). Skills development models and simulations develop individual and team skills. An example of this type of simulation is the Observed Fire Trainer for Bradley Infantry Fighting Vehicle or M1 tank crews. Exercise drivers are combat simulations that resolve battles based on inputs provided by the target commanders and staffs. They frequently allow real time interaction and "create some of the stress, confusion, and time pressure of battle along with the simulated combat scenario" (13:14). In actuality, they can be thought of as "group skills" development models. JANUS is a training model.

The original SPARTAN lies somewhere between both of these classifications. It was designed along the lines of an analytic model and is used as a skills developer. It demonstrates the form and function of an analytic model without the output of such a model. It does not support any training, other than skills development of military analysts.

Classification by Model Qualities. Model qualities can be thought of as the real entities and processes that the model represents (1:7). Qualities include the physical space in which entities and processes operate and the categories of weapons that the model represents. Another quality is the level of detail of processes and entities.

Both JANUS and CASTFOREM operate in a three dimensional battlefield. They represent both ground and air conflict. They also both represent combined arms task forces. A combined arms task force is a battalion (infantry or armor) that has been augmented with other types of forces. A battalion could number over 1200 men, 60 armored fighting vehicles, 200 support and logistics vehicles and aircraft. Thus, JANUS and CASTFOREM model tanks, infantry fighting vehicles (IFVs), artillery, close air (to a limited extent), infantry, military intelligence, air defense artillery, engineers and other elements (7:1-3,8:A1-A4). The mission area of both of these models is the close fight between battalion or brigade sized units. Both JANUS and CASTFOREM model processes and entities at the individual weapons system level. That is, they model individual tanks, IFVs, and dismounted soldiers.

SPARTAN also operates in a three dimensional battlefield, but only ground conflict is represented. In fact, SPARTAN models only dismounted infantry engagements. The only weapons that SPARTAN models are direct fire semiautomatic rifles. There are four types of weapons, differentiated by their accuracy at different ranges. The mission area for SPARTAN is the close fight between squad sized (approximately 6-9 men) elements.

Another quality that can be used to classify models is the environment. The environment of the model is the state of the physical system in which the model operates. Environment includes terrain, weather, light (day/night), built-up areas and sea states (1:7). CASTFOREM models terrain and vegetation, static weather (does not change throughout the battle), dynamic obscurants like smoke and dust, and chemical contaminants (7:1). JANUS also plays these systems (8:1-50). SPARTAN's environment is simple, it models only terrain and does not represent effects from weather, obscurants, buildings, temperature, ambient light levels, electric warfare, or Nuclear, Biological or Chemical effects.

Force composition is another quality of models that can be used to classify them. Force composition is the mix of forces that is portrayed by the model. As was stated above, both JANUS and CASTFOREM model combined arms task forces, either battalion or brigade sized. The number of entities and systems this represents is enormous. SPARTAN attempts only a limited number of entities. It only allows a maximum of 12 soldiers (combined total from both sides). No headquarters are modeled, although one soldier on either side can be designated as squad leader.

To model these entities, CPT Cox used 17 attributes (5:47):

Table 1 Attributes of SPARTAN's Entities

ATTRIBUTE	DESCRIPTION
1	Soldier's current horizontal coordinate
2	Soldier's current vertical coordinate
3	Soldier's current elevation
4	Soldier's last horizontal coordinate
5	Soldier's last vertical coordinate
6	Soldier's height
7	Soldier's speed (not adjusted for posture)
8	Soldier's current direction of travel
9	Soldier's move status (moving/not moving)
10	Soldier's weapon type
11	Soldier's current ammunition count
12	Soldier's current wound status
13	Soldier's posture
14	Squad leader marker
15	Soldier's original attack direction
16	Next target to engage
17	Side (RED/BLUE)

<u>Classification by Construction</u>. Another way to classify models is by model construction. Construction refers to the design of the model (1:9). Elements of construction include the amount and type of human interaction, how the model handles time processing, how the model represents randomness, and the sidedness of the model.

Human participation is the extent "to which human presence is allowed or required to influence the operation of the model" (1:9). CASTFOREM and JANUS(T) differ in this respect. Once CASTFOREM has begun a model run, no human input is required. CASTFOREM has an expert system that uses decision tables, based on the input scenario, to make all required battlefield decisions (7:2,18-23). JANUS(T) does not have an expert system; human participation is required to make decisions when the model is used as a training tool. SPARTAN resembles CASTFOREM in that no human participation is required (or allowed) once model runs begin.

Another design quality is how the model treats time. There are two basic types of time processing, static and dynamic. Static models do not consider changes to the system as a function of time and thus do not represent time (14:A-7). Dynamic models, on the other hand, explicitly model the impact of time on system state changes. CASTFOREM, JANUS and SPARTAN are all dynamic models.

Dynamic models can further be broken down into time step, event step, or closed form solution models (1:9). Neither JANUS, CASTFOREM, or SPARTAN are closed form models, which are models in the form of a set of differential equations which have a closed form solution (14:A-7). Most high resolution combat models are either time step or event step models. In time step models, the model time keeping system advances at a set discrete amount. At each time change, the states of the system are updated based on the new time (1:7,13:2-6). In event step modeling, as the model finishes an activity, it checks to see what the next scheduled event is. The model checks the time of that scheduled event and advances the model clock to that time (14:A-7,13:1-3,2-1). There are also hybrid time advance mechanisms, where the model clock slows to match real time advancement. SPARTAN, CASTFOREM, and JANUS are all event step models, although CASTFOREM does allow time step processing in some cases (14:C1).

Another aspect of model construction is how the model treats randomness. This is important to combat models because what is normally called the fog of war is actually the uncertainty of battle because of random outcomes to events on the battlefield (usually because of human decisions) and a lack of perfect information. There are two basic approaches to treating randomness, models either ignore it, or consider it in some manner (14:A-10). Deterministic models ignore randomness. Some processes do not have randomness, in which case solving them deterministically makes sense. Most military processes are stochastic, not deterministic, however, and models must deal with this property. Some models (expected value models) try to do this deterministically, they ignore "the inherent randomness in the stochastic processes by replacing all random variables with deterministic quantities (the expected outcome) of the process" (14:A9).

Most high resolution combat models use one of two ways (or a combination thereof) to treat the stochastic nature of combat. One method is known as the Monte Carlo method (defined in Chapter I) (1:2-46;14:A-10). The other possible way to determine the outcome is to calculate the results directly. This may give more accurate results, but can be costly in terms of computational time. JANUS and CASTFOREM use both the Monte Carlo method and the direct computational method in different parts of the programs.

SPARTAN treats probabilities of acquisition, detection, hit, results of a hit, reaction to fire, and time till the next event using the Monte Carlo method and a Uniform (0,1) random number. Line of Sight is deterministically computed.

Another way to classify models by construction is by sidedness. A side in a combat model is a collection of entities and resources used in cooperation to achieve a common goal (14:A-15). CASTFOREM, JANUS, and SPARTAN are all two-sided models. They all have two sides that are in conflict with each other. All three models are also symmetric, both sides in the models having the same relative resources and the ability to employ them to some varying degree of effectiveness.

SPARTAN's Scenario

The scenario is one of the most important aspects of the simulation. It establishes the environment of the conflict, the terrain aspects, the force composition, the systems played and the tactics involved. SPARTAN requires no written scenario, mainly because the user establishes the scenario when he runs the start-up program. In SPARTAN the scenario includes terrain elevation and mobility, number of red or blue soldiers, position, direction of movement, weapon type, speed, posture or any other soldier attributes the user might alter. The user can also establish the initial events the model will execute.

Data Base Information

The Data Base information required for SPARTAN is very small. SPARTAN requires only six look-up tables. There is a probability of acquisition table (based on observer-target range and target posture), a probability of detection table (based on observer-target range, target posture, and time duration of observation), and four probability of hit tables (based on weapon type, range, and target posture). These tables are not accessible to SPARTAN users and the only way to change them is to reconstruct Cox's method of calculating them and recompute them.

Terrain Functions

The level of detail in modeling terrain should be consistent with the level of detail of the processes that the model replicates (13:1-12). In line with this fact, both JANUS and CASTFOREM model terrain in great detail. Their terrain is digitized and has the attributes of trafficability, elevation, and clutter (representing vegetation), and includes such features as rivers. SPARTAN, however, was not intended to offer such detail, but rather offer a broad outline of what high resolution combat models do. Therefore SPARTAN only represents elevation and mobility.

All three models use a grid terrain model to represent terrain. In a grid terrain model, a grid of regular polygonal grid cells is overlaid across the battlefield. Each of these cells has its own attributes such as clutter factor, elevation, and trafficability factors (13:3-1,3-5). CASTFOREM usually uses 100m squares with uniform attributes throughout the square (7:13), although squares can be altered in size depending on the scenario. JANUS also uses squares (again size can vary)(8:33). In JANUS(T), attributes apply uniformly through each square except for the elevation. The elevation attribute marks the lower left corner of each square, elevation changes proportionally to the distance moved from the marking corner and towards the marking corners of neighboring squares.

SPARTAN divides its one kilometer square battlefield into 2500 squares that are 20 meters on a side. Elevation and mobility factors are uniform throughout each square. Only the mobility factor has an impact on movement, soldiers do not slow for changes in slope. Elevation affects line of sight and thus, target detection and engagement.

Combat Processes

The Army's FM 100-5 OPERATIONS has broken the battlefield elements into seven battlefield operating systems (BOS). These are:

1) Maneuver

2) Fire Support

3) Air Defense

4) Mobility/Countermobility/Survivability

5) Combat Service Support

6) Intelligence/Electronic warfare

7) Command and Control

The BOS all support one or more of the fundamentals of combat, "shoot, move, and communicate". Models that replicate combat must simulate one or more of these fundamentals. These fundamentals are usually modeled by further breaking them down into multiple components known as combat processes. The original SPARTAN replicated the following combat processes:

1) Movement

2) Target Search

3) Target Selection

4) Direct Fire Engagement

5) Command and Control

Movement. Although movement is in reality a continuous function, most high resolution models represent it through a continuous series of discrete steps. Both JANUS and CASTFOREM model movement this way. In both models movement is affected by entity attributes such as speed, posture, and mobility and by terrain attributes such as trafficability, vegetation, and slope. Both models also use maneuver control points (8:277,7:140). Prior to model runs (or, for JANUS(T), during model execution), routes are selected for entities and maneuver control points designated along these routes to control movement. Entities then move from point to point along the route. JANUS(T) attempts to move entities 50 meters per move (modified by terrain and obstacles). It then computes the time that the move would have taken, and schedules the next move for that entity at that time (8:411-412). CASTFOREM also attempts to move entities set distances, but it is not locked into a set sequence of movement control points. It uses the Dykstra "shortest Path" algorithm to compute routes (7:142). CASTFOREM can also adjust speed to maintain formation within units.

CPT Cox also designed SPARTAN to represent movement in discrete steps. SPARTAN's movements were in 20 meter increments, with the new location being computed by the equations below (5:70):

$$xnew=xold+20\times COS(DIR) \tag{1}$$

$$ynew=yold+20\times SIN(DIR) \tag{2}$$

Where DIR is the attribute of each soldier that denotes direction of travel.

CPT Cox used a method similar to that of JANUS by making the time to move to the new position a random variable. His computation of movement time involved the speed of the soldiers (fixed at 20 meters per move), the mobility factor of the terrain cell in which the soldier started his move, and the soldier's posture. Equation (3) shows his move time calculation (5:72).

$$movetime = time + \frac{RND(0,1) \times 100}{speed \times mobfactor \times posture} \quad (3)$$

SPARTAN used two different routines to control movement. In the subroutine STARTMOVE, SPARTAN checked to make sure the soldier was in a move status. If not, the subroutine ended and the next event on the event schedule was called. If the soldier was in a move status, the new position was computed, the graphics updated, and the time to complete the move computed. Then a call to the subroutine ENDMOVE was scheduled. At ENDMOVE, a random variable from a triangular distribution was used to determine the time to the next STARTMOVE. A STARTMOVE was then scheduled for that time (5:69-74).

CPT Cox acknowledged the following limitations inherent in the low level of detail in his movement process (5:72-74):

1) Entities move and stop in discrete steps rather than move continuously.

2) Entities do not have the option of moving less than 20 meters, regardless of the situation.

3) Terrain slope does not affect movement. (This is a realistic assumption given the tactical situation and the nature of the entities.)

4) The mobility factor of the move is constant throughout the move, regardless of whether or not it changed.

5) Entity posture does not change after ENDMOVE and before STARTMOVE.

6) Soldier fatigue is not a factor (A realistic assumption in light of the short battle duration).

7) Movement is scaled for screen graphics.

8) Posture changes are instantaneous and do not affect a move in progress.

Target Search. Target search is one of the most important and most difficult aspects of combat to model. It includes aspects of acquisition and detection. Detection is defined as the "event constituted by the observer's becoming aware of the presence and possibly of the position and even in some cases the motion of the target" (13:4-1). There are several levels of target acquisition within the above definition of detection. These are (13:4-1-4-2):

1) Cuing information. This provides the observer the approximate location for further search.

2) Detection. The observer decides that an object in his field has military interest.

3) Classification. The observer is able to distinguish broad target categories.

4) Recognition. The observer is able to discriminate between finer classes of targets.

5) Identification. The observer precisely identifies the target.

When attempting to model target acquisition and detection, two significant characteristics about detection must be taken into account. First, three physical conditions must exist: the observer must have line-of-sight to the target (there must be no obstructions blocking his view of the target), the target must have a physical signature that the observer can detect, and the observer must be looking in the right direction. The second characteristic is that even if all three physical conditions exist, there is no guarantee that the observer will detect the target (13:4-3). Thus, search processes involve checks for line of sight, checks to see if the target offers a distinguishable signature, and checks to see if the observer is looking at the target long enough to acquire it.

Line of Sight. Line of sight (LOS) refers to observer having an unobstructed view of the target. LOS can be blocked by intervening terrain or vegetation. It generally "considers only the major land forms of the terrain along with major vegetation" (13:3-3). In general, LOS checks consist of determining the line between the observers eye and some target height. Checks are made along this line to determine if terrain or vegetation blocks it (13:3-4).

Both JANUS and CASTFOREM determine line of sight deterministically. In JANUS(T), the range from observer to target is determined. The range is then divided by the largest dimension (called d) of the uniform sized terrain polygon. This result gives the number of terrain elevation checks that JANUS conducts. The line of sight is then checked every d distance. If the terrain does not block the LOS, JANUS checks vegetation factors and degrades the LOS by a percentage. If the remaining percentage does not fall below some threshold, LOS exists (8:348-351). CASTFOREM measures LOS differently than JANUS. It does not measure straight LOS. Its LOS moves along vertical or horizontal axes (like a step function). "This method is faster than the traditional straight line methodology and has been shown to statistically agree with it, when the grid square resolution is at least 50 meters per side" (7:6).

SPARTAN's method of LOS checks is much like that of JANUS. SPARTAN determines the equation of the line between the eye of the observer and the top of the target. Terrain elevation is then checked every 10 meters along this line to see if the LOS is blocked. Partial obscuration is not counted, if the top of the target's head can be seen, then the whole target is counted as being seen (5:75-77). Since vegetation is not modeled, it has no impact on LOS. The disadvantage to this method is that it is possible for the observer-target line to cross part of an intervening terrain cell with blocking elevation and not fail LOS, because this terrain cell falls between 10 meter checks.

Target Acquisition. If LOS exists between observer and target, the model must determine if the observer detects the target. Both JANUS and CASTFOREM model target acquisition/detection using the Night Vision Electro-Optical Laboratory (NVEOL) detection model. The algorithms used in the NVEOL model take into account target dimensions, observer-target range, target-background contrast, sky-ground brightness ratio, and visual attenuation caused by atmospheric conditions (8:352-365). The drawback to the NVEOL model is that the data for it was based on stationary observers looking at nonfiring stationary targets. Therefore heuristics must be used to deal with targets that are shooting or moving and observers that are moving. Also, the NVEOL model deals only with randomly selected observer-target pairs (17:26).

The NVEOL model is based on the idea of how many resolvable cycles an observer can detect on a potential target. A cycle is a pattern of light and dark bars of the same width of the minimum dimension of the target. The contrast of the light and dark bars is the same as the contrast of the target and its background (17:25). Using the ratio of number of cycles the observer can detect to some threshold level for level of detection, NVEOL computes the probability of detection.

The first step of the NVEOL model is the computation of the attenuated contrast. This is the apparent target-background contrast, after sky-ground brightness and atmospheric attenuation has been computed (17:25).

$$attenuatedcontrast=\frac{targetcontrast}{1+SOG\times(e^{vis\times range}-1)} \qquad (4)$$

SOG= Sky-ground brightness = 2.5 on a bright day
vis = atmospheric attenuation coefficient
range = observer-target range in kilometers
target contrast = .2--.3 for visual targets

The attenuated contrast is then applied to a sixth degree polynomial (in natural log) and multiplied by the width of the target in milliradians (width of target in meters divided by range in kilometers). This result is the number of cycles the observer is able to detect across the target (8:352-365).

The probability of detection in NVEOL is based on two probabilities: the probability that the observer will detect the target given unlimited time (called Pinf or P1) and the probability that detection is made at some time given the target is detectable and the observer is looking at the target (called Pfov or P2) (8:352-365). The calculation for P1 is (17:26-27)

$$P1=\frac{TERM}{1+TERM} \tag{5}$$

where TERM = cycle ratio ^power
cycle ratio = $\frac{\text{cycles resolved by observer}}{\text{cycles for 50}}$
power = 2.7 + .7 * cycle ratio

Cycles for 50 is defined as the number of cycles required for 50% of the population to detect the target. Cycles for 50 for various detection levels are listed below (17:27):

Simple detection:	1 - 2
Recognition:	3 - 5
Classification:	6

The equation for P2 is (17:26)

$$P2=1-e^{-k\times t\times CYCLERATIO} \tag{6}$$

k = constant = 1/6.8
t = the time spent looking at some particular field of view.

The overall probability of detection is a function of P1 and P2

$$Pr(DETECT)=P1\times P2 \tag{7}$$

Both JANUS and CASTFOREM use the NVEOL model in similar fashions. During model start-up, every observer-target pair is assigned a random threshold P1 level (8:352-365). This has the affect of making the possibility of detection random (some observers need to see more of a target than others before they can detect it and observers will need to see different amounts of the same types of targets to detect them). Then, as target detection/acquisition routines are employed, P1 is computed deterministically for each target in the observer's search field. This is compared against the threshold level. If his P1 is larger than the threshold, a random search time for the field of view containing the target is drawn and P2 is determined.

P2 is determined by the field of search and by the random time spent in each part of the field of search. CASTFOREM always conducts searches of 360 degrees. JANUS uses a full 360 degree search when entities are moving, but the field is cut to 180 degrees when the entity stops. Once P2 is determined, the overall P(detection) is computed. A random number draw then decides the success or failure of detection (8:352-365;7:7). Once a target is detected, both JANUS and CASTFOREM retain it on the potential target list unless the observer loses line of sight.

CPT Cox employed a simplified version of the NVEOL model in SPARTAN. His method was proposed by Bailey, a contractor hired to simplify the search routines of JANUS and CASTFOREM (5:79). The proposed routine was never implemented. Bailey's algorithm also employs two probabilities: one probability (P1) is the probability of eventual detection given unlimited observation time (5:79) and the other probability (P2) is the probability of detection given the observer is looking at a certain field of view for some random time (5:82).

$$P1=1-e^{-.84\times(\frac{C}{M})^{2.4}} \qquad (8)$$

where C = target height/range to target
M = 3.5
-.84 is a scaling factor added since no empirical data was used on the sensory capabilities for human eyesight.

$$P2=1-e^{-\frac{C}{M}\times\frac{t}{6.8}} \qquad (9)$$

where C and M are the same as above
t is a random variable ranging from (.4 - 4)

CPT Cox used MATHCAD to prepare a table for each probability based on target posture and in ranges of 100 increments. The table for P2 also moves time in increments of .4 time units.

When SPARTAN called a search routine, the P1 for the target range and posture was obtained from the table and compared to an arbitrary threshold of .2. If it was possible for the observer to detect the target signature, a random time was drawn (between .4 and 4 time units) and a P2 drawn from the table. A Monte Carlo trial then determined if the target was to be added to the potential target list. CPT Cox's search process was memoryless. Each time the entity went to the search routine, it was as if the entity had never detected anyone, so his potential target list was empty. SPARTAN's search replicated a full 360 degree field of search. Also like JANUS and CASTFOREM, SPARTAN only sought to identify foes, friendly soldiers were not searched for and identified (therefor there was no fratricide replication).

Target Selection. Once an entity has detected a potential target, it must decide whether or not to engage that target and with what weapon. If more than one target is detected, then the entity must choose which target or targets to engage. This is the purpose of target selection. The problem with target selection is that there is "no basic seminal theory" telling how to accomplish it (13:6-1). One way to model target selection is the DYNTACS adjusted range formula (13:6-3). In this method, targets are weighted by their attributes, recent actions such as whether the target has fired recently, whether the target is under fire from another friendly element and whether the target was in the observer's sector of responsibility. Another method is for target selection to be decided by user input priorities (13:6-4).

JANUS and CASTFOREM use dissimilar algorithms for target selection. CASTFOREM uses a method similar to the DYNTACS formula (7:11). JANUS(T) selects targets based on the Single Shot Probability of Kill (SSPK). The first step in the JANUS process is a check to make sure the target is within range of the primary weapon system. If the target is within range, the observer checks remaining ammunition of his primary weapon. If he is out of ammunition, he attempts target selection based on his secondary weapons system. Once the type of weapon has been decided, the SSPK of all targets acquired by the observer is determined. ~~The~~ SSPK is based on range, movement status of both target and observer and protection factor of target. If the SSPK is larger than a threshold level of .05, target engagement begins (8:371-372).

SPARTAN uses a method similar to JANUS. Once an entity has completed a search cycle and has one or more potential targets, the SELECT subroutine is scheduled. In the select subroutine, all the P2 values for acquisition are summed up. If the sum is greater than an arbitrary value of .2, selection continues. If not, a search is scheduled. If selection continues, the P2 values are normalized so they sum to one. A random number is then drawn to determine which target the observer will engage and an engagement is scheduled. This target is then carried as an attribute on the observer's attribute list (5:83-85).

Cox's use of probability of detection to determine target selection intuitively seems to make sense in the absence of rules of engagement, target priorities or sectors of fire. Although he did not modify target detection based on signatures (such as firing or movement), it seems logical to engage the target that attracted most of the shooter's attention.

Target Engagement. There are two basic types of engagement: direct fire and indirect fire. In direct fire engagements, observers have line of sight to their targets and aim directly at them. In indirect fire engagements, gunners do not have a view of the target and are aiming at some coordinates, usually forwarded to them by someone who has acquired the target either visually or by electronic means.

Direct Fire. All direct fire engagements are modeled in much the same manner. Before firing, the observer checks to see if LOS to his target still exists. If LOS exists, the model performs a Monte Carlo experiment to see if the shot (or shots) hit or miss the target. In JANUS, the SSPK determines whether or not the round impacts. CASTFOREM uses "a normal bivariate distribution with a bias off the aimpoint and dispersion" and draws a random impact point to determine whether the round hits the target (5:19).

When SPARTAN's direct fire subroutine is called, it first checks to see if LOS still exists. If LOS exists, the entity is checked for ammunition. If he has ammunition, the ammunition count is decremented by one and a look up table is accessed for the probability of hit for the target range, posture and weapon type (5:83-87).

Cox used the equation below to calculate his P(hit) (5:87).

$$Pr(hit)=1-e^{\frac{R^2}{2xvar}} \qquad (10)$$

where $R^2=x^2+y^2$

var = aim and ballistic error of round in both x and y directions.

This formula assumes a circular target and does not take target aspect angle into account. The P(hit) tables were adjusted for target postures by adjusting R by a percentage based on whether the target was prone or crouching.

Once the P(hit) was determined, a random Uniform(0,1) variable was drawn to determine whether the round hit or not.

<u>Indirect Fire</u>. Indirect fire is more complicated than direct fire because artillery is an area effects weapon, not a point effects weapon (excluding guided rounds like Copperhead). P(hit) is based on number of rounds fired, center of impact of rounds (center of sheaf), size and aspect of sheaf, type of rounds (high explosive or Improved Conventional Munitions), point of detonation (air, ground, below ground) and protection factor of target.

JANUS incorporates the following into calculating P(hit) of artillery (8:378):

1) Round to round ballistic errors.
2) Artillery formation and distance between firing unit elements.
3) Parallel or converging sheaf.

When all rounds' impacts have been located, JANUS determines a maximum effects box based on the imaginary box that contains all the rounds, plus an added factor based on distance determined by target protection factor. In addition, a suppression box is drawn 200 meters around the effects box (8:378). CASTFOREM determines indirect fire engagement results much the same way (6:104-113).

The original SPARTAN does not model indirect fire engagements.

Impact Assessment. If the result of an engagement is determined to be a hit, then an assessment of the damage must be accomplished. Lethality models used in high resolution combat simulations, simulate each round individually (13:8-1). In some cases, such as when the weapon is a machine gun, this round is actually a burst of several individual bullets. Machine guns and automatic cannon are modeled like this because the extreme resolution needed to model each bullet separately "would be prohibitive for most force on force modelling purposes" (13:9-1).

There are basically two types of projectiles: impact projectiles, which must hit a target to cause damage, and fragmenting projectiles, which explode and cause damage either by the explosion or by fragments (13:8-1).

For impact projectiles, a Bernoulli trial is compared to the SSPK to determine the level of damage to the target. In JANUS, elements of the target are either killed or not. CASTFOREM is a bit more detailed, computing the exact point of impact and determining the level of damage, either firepower kill, mobility kill, mobility and firepower kill, or catastrophic kill (7:148).

Given that the round is a hit, SPARTAN uses a Uniform (0,1) random variable to determine if the target was killed or wounded. Targets have a 30% chance of being killed and a 70% chance of being wounded. Wounds are not cumulative; being wounded does not increase the target's chance of being killed if hit again. Also, the target's posture and move status do not change upon being wounded.

There are two different methods of calculating damage from fragmenting projectiles: the cookie cutter method and the Carleton damage function. JANUS can use either method (8:378-379). CASTFOREM uses the Carleton function (7:150).

The cookie cutter method makes use of the fragment sheaf discussed above. This sheaf is a function of sheaf spread, round burst radius, and impact dispersion. If the target is determined to be outside of the sheaf, then no damage is assessed.

The Carleton function computes P(hit) as a function of the ratio of the miss area divided by the bursting area (7:151, 17:78).

$$Pr(hit) = e^{\frac{-Missdistance^2}{burstradius^2}} \tag{11}$$

A Bernoulli trial then decides whether the target is hit.

<u>React to Fire</u>. When a target is taken under fire, and detects that it is under fire, it will normally take some action. This action could be to seek cover from the fire,

move to evade the fire, attempt to locate the source of the fire and fire back, or attempt to fire in the perceived direction of the fire and suppress the attacker. This is known as reaction to fire.

Regardless of the reaction, the effect of fire is suppression. This means that the target under fire suffers some degradation of performance because of its actions as a result of fire. JANUS only models suppression as a result of indirect fire. The effects are that the target can move, but not shoot. CASTFOREM models suppression and reaction to fire for both direct and indirect fire. If the target has sustained less than a catastrophic kill, decision tables decide the appropriate action for the survivors to take (7:150).

SPARTAN models suppression by giving targets three possible reactions to receiving fire. The target has 40% chance of moving to a prone position and slowing his speed. This models suppression as the prone posture inhibits acquisition. The entity also has a 20% chance of reversing directions and moving 20 meters. Finally, the target has a 40% chance of ignoring the fire (5:91-92).

Command and Control. Command and control is the ability of the decision makers in the model to influence the battle. CASTFOREM, being non-interactive, models command and control through myriads of decision tables and some expert logic. Based on changing situations the model checks the decision tables and alters instructions to subordinate entities (7:17-42). JANUS, being interactive, does not model command and control, depending on the players to make decisions.

SPARTAN has a limited command and control function. One entity on either side can be designated as squad leader. If a squad leader is designated, all other entities on his side will reorient to engage targets that he engages. If he dies, then the other entities reorient on their original direction of movement (5:91).

<u>Output</u>

The type of output desired from the model should drive model design. Simulations designed for training should provide output that provides some indication of the success of the trainee in meeting performance measures of the task on which he is being trained. Simulations for analysis should produce output that facilitates the analysis being conducted.

JANUS' postprocessor produces up to 11 reports. These are (8:323-345):

1) Artillery Fire Report
2) Artillery Summary
3) Direct Fire Report
4) Coroner's Report
5) Killer Victim Scorecard
6) Heat and Chemical Casualties
7) Temperature and Workload Profiles

8) Minefields and Crossers

9) Engagement Range Analysis

10) Force Loss Analysis

11) Detections

Some of these are analyses of battle results which are left over from JANUS' roots as an analytical tool and are of limited value. The history files are useful for After Action Reviews, as the training audience tries to see what they did and when.

SPARTAN's output is very similar to JANUS'. At the end of every model run, SPARTAN asks users if they desire to review the four output files. The first file is a final score card detailing start and end strengths, ammunition remaining, number of hits on either side, and number of wounded. The second output file is the final attribute list for either side's soldiers. The third output file is the final potential target table with final probability of detection values. SPARTAN also produces a fourth, history, file that lists event type, event time and event actor.

Summary

The original version of SPARTAN is a high resolution, two sided, force-on-force, stochastic, event sequenced simulation of dismounted infantry conflict. It models movement, target search, target selection, target engagement and some limited command and control processes. The methodologies used to model these processes were derived from those used in the current generation of high resolution land combat models. On the whole, SPARTAN faithfully represented current modeling methods, but there were some areas that needed improvement. The next chapter discusses how I planned to implement these improvements.

III. Model Development Process

Introduction

This chapter will describe the general model building process used to create and refine SPARTAN. The first section outlines the general methodology used in creating SPARTAN. The next section defines the modeling problem, model objectives, and some assumptions about the objectives. This is followed by a discussion about model formulation which specifically addresses SPARTAN's modeling environment and model components. The next section is a discussion about model development, focusing on creation of the data base, event set management, model enrichment, treatment of randomness, and development of instructional components. The last section discusses SPARTAN's model assessment processes.

Development Methodology

Although SPARTAN was already designed and coded, the fact that I did not have access to the original uncompiled programs and the numerous revisions to methodology and changes I made, make the SPARTAN essentially a new program. The overall process required to build this new SPARTAN is flowcharted in Figure 1.

The framework for simulation development applied to SPARTAN was a combination of the simulation process

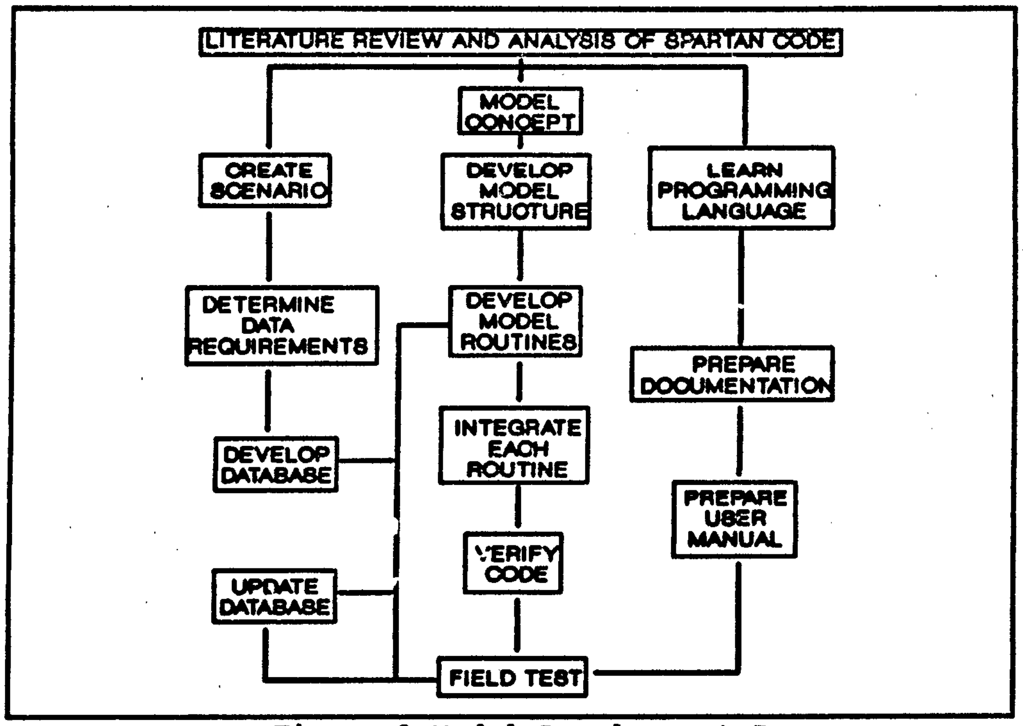


Figure 1 Model Development Process

described by Pritsker (19:11-12) and the conical methodology described by Richard Nance (18:38-43).

Pritsker lists 10 steps of simulation development (see Chapter 1), six of which were applicable for SPARTAN development (19:11-12):

1) Problem formulation. The definition of the problem to be studied, including a statement of the problem-solving objective.

2) Model Building. The abstraction of the system into mathematical-logical relationships in accordance with the problem formulation.

3) Data Acquisition. The identification, specification, and collection of data.

4) Model Translation. The preparation of the model for computer processing.

5) Verification. The process of establishing that the computer program executes as intended.

6) Validation. The process of establishing that at desired accuracy or correspondence exists between the simulation model and the real system.

The conical methodology essentially subdivides Pritsker's steps one and two into substeps, more clearly defining problem definition and model building. An extracted outline of the conical methodology is presented in Figure 2.

I. Statement of the study objective

   A. Problem Statement
   B. Objectives and assumptions about objectives

II. Modeling Environment

   A. Modeling effort available/required
   B. Modeling Assumptions

      1. Boundaries
      2. Interaction with environment

III. Model Description

   A. Identify the objects and their attributes
   B. Submodels with possible sublevels

VI. Model Validation and Verification Procedures

   A. Validation tests
   B. Verification criteria and tests

V. Model experimentation

Figure 2 Conical Methodology Outline

Problem Definition

The first task of both model development methodologies is a determination of the study problem and an "explicit statement of the objectives of the analysis" (19:11). In this instance, the problem was identified in Chapter II as "to continue development of SPARTAN, and improve its value as an instructional tool for high resolution land combat modeling courses".

Once the problem is clearly defined, the modeler must decide on the objectives of the simulation. Having the problem of continuing development of SPARTAN, I decided upon the objectives enumerated in Chapter I: portability, simplicity, applicability, and useability.

Portability is the degree to which SPARTAN can be transferred from computer to computer. This means that SPARTAN must be small enough to be carried on a single disc and written in code readily available to most users. With this objective in mind, I decided that SPARTAN must be capable of running on all IBM XT compatible computers (512K) with EGA minimum graphics ability. For this reason, SPARTAN was written in QuickBASIC, and complied into an executable (*.exe) file.

Simplicity is the degree to which the workings of SPARTAN can be readily followed and understood by the user. This is mainly because the intended audience of SPARTAN is the beginning combat modeler, who may have only limited knowledge of combat processes or computers. For this reason, the number of processes and entities involved in SPARTAN was kept to a minimum. Also, simplicity eliminated the necessity for excruciatingly detailed or extraordinarily accurate replications of reality and allowed simplifying assumptions about data and processes. The objective of simplicity also allowed me to eliminate the option of human input during simulation runs.

Applicability means that SPARTAN must accurately portray processes like those found in the current generation of high resolution land combat models. Given the limitations in computing power caused by the first objective, SPARTAN could not and did not need to completely recreate JANUS and CASTFOREM. It did need to accurately represent the techniques used by those models in such a way that novice modelers could understand them.

Usability is a combination of the other three objectives. Usability means that SPARTAN can meet its intended function of teaching modelers about high resolution land combat modeling. The model must allow users to alter inputs and conduct sensitivity analysis (as is done in "real" models). SPARTAN also required extensive documentation so that users can grasp what the model was doing and how the model was doing it. The graphical representation of model activities had to enhance, not complicate, understanding of model processes. SPARTAN also had to allow the user to query it about the current state of the system. The model had to be capable of producing output about the ending system state. Lastly, the model had to run faster than real time so that replications could be easily done.

Model Formulation

Having identified the problem and model objectives and assumptions about those objectives, the next step of the model building process was the formulation of the model. This included identifying the modeling environment and the model definitions and developing the model itself.

Modeling Environment. The first step of model formulation is identifying the modeling environment. The modeling environment includes the modeling effort and modeling assumptions.

Modeling Effort. Assessing the modeling effort is frequently the most overlooked and forgotten aspect of the modeling environment. Because of the nature of SPARTAN's intended purpose, this assessment was not so much an assessment of what measures of effectiveness SPARTAN was required to produce, but rather an assessment of the time limitation to finishing SPARTAN. SPARTAN was required to be completed within 26 weeks.

Modeling Assumptions. This assessment, like that of modeling effort, was an effort to more concretely identify the limits of model construction and thus facilitate model formulation. The list below is not comprehensive, but does enumerate the major limiting assumptions I made in formulating SPARTAN:

1) Model Type. Because SPARTAN is designed for instructing land combat modeling students, not training experienced soldiers, no interaction with the simulation during model runs was required. (The model is not a man-in-the-loop). For

this reason, SPARTAN was designed like a analytical model, not a training model.

2) Model Domain. The model dom 'n is the "physical or abstract space in which the entities and processes operate" (1:7). In keeping with the objective of simplicity, I assumed SPARTAN's domain would be land only.

3) Model Span. SPAN is the scale of the domain. SPARTAN's domain is local (as opposed to global or theater). The area modeled is a one square kilometer area.

4) Boundaries. The boundaries of SPARTAN are really the scope of the modeled conflict. SPARTAN is primarily a attrition model of small unit conflict. Other than some indirect fire, no off screen factors influence simulation outcomes.

5) Level of detail. In SPARTAN all events are decided at the individual soldier level. No processes are aggregated.

Model Definition. Once the modeling effort and modeling assumptions have been identified and the scope of the model narrowed, the model must be defined. Models of systems have "both a static and a dynamic description" (19:11). The static description defines the objects or entities of the system and their characteristics. The dynamic description defines how the objects and entities interact to cause system state changes (19:11-12).

Static Description. The first step in model definition is the decision of what objects can either change the model state or cause some action to occur (5:45). Like the previous version of SPARTAN, the objects and entities are:

1) Terrain cells--The data records for terrain representation. Since the 1 km square

area is represented by square cells 20 meters on a side, there are 2500 cells.

2) Events--The records controlling actions of the model.

3) Soldiers--The operational entities. A maximum of 12 soldiers are allowed.

Each of these objects is characterized by attributes, which more fully describe the capabilities or type of object. Determining the number of attributes each object required was a function of the model objectives. The number had to be sufficient to meet these objectives without being superfluous. Objects' attributes are listed in Tables 2-4.

Table 2 Terrain Attributes

ATTRIBUTE	DESCRIPTION	RANGE OF VALUES
1	Mobility factor	.1 - 1.0
2	Background contrast	.5 - 1.0
3	Elevation	60m - 110m

Table 3 Event Attributes

ATTRIBUTE	DESCRIPTION	RANGE OF VALUES
1	Event type	1 - 9
2	Event actor	1 - 12
3	Event time	0.0 - 9999.0

The system also has attributes. Like the attributes of objects, system attributes describe the characteristics and state of the system. SPARTAN's attributes are listed in Table 5.

Dynamic Description. The dynamic description defines how the objects of the system interact to change the system state. In SPARTAN, these activities fall into three

Table 4 Soldier Attributes

ATTRIBUTE	DESCRIPTION	RANGE OF VALUES
1	Side	1 = BLUE, -1 = RED
2	Duty position	1=SL, 2=ASL, 3=GRNDR 4=AR, 5=Rifleman
3	Horizontal coord.	0.1 - 999.99
4	Vertical coord.	0.1 - 999.99
5	Number of Grenades	0 - 32
6	Time last fired wpn	0.0 - 9999.99
7	Posture prior to direct fire engagement	
8	Movement Direction	0.0 - 6.28 radians
9	Movement status	1=moving, 0=not moving
10	posture	1=standing, 2=crouching 3= prone
11	Weapon type	1=M16A2, 2=AK74, 3=SAW 4=M203, 5&6=user defined
12	Rounds in magazine	M16=30, AK74=40, SAW=200
13	Number of magazines	M16 & AK74 = 6 SAW = 3
14	Selected target	0 - 12
15	Wound status	2 =alive, 1 =wounded 0 =dead

Table 5 System Attributes

VARIABLE NAME	DESCRIPTION
time	Simulation clock
bluecount	Current number of BLUE alive
redcount	Current number of RED alive
termevnt	number of events processed

categories. The first category consists of those activities that are assumed to occur instantaneously, meaning that they do not cause the simulation clock to advance. These events are not scheduled activities and are not put on the event list. They are usually sub-activities of scheduled events and are used to update or alter object and system attributes. These activities are listed below:

Instantaneous Activities

Line of sight (LOS): This activity checks for intervening terrain or vegetation between observers and potential targets. There is no partial line of sight, it either exists or it does not.

Detecting obstacles

Determining target posture (and thus size)

Determining observer-target range

Determining probability of detection

Developing target lists

Decrementing ammunition counts

Determining probability of hit

Determining round impact outcomes

Plotting entities on screen

Plotting engagements on screen

Changing soldier posture and direction

Calling indirect fire

Intra-squad communication

The second type of activity is the time duration activity. These activities require time to accomplish and are thus scheduled on the event calendar. In SPARTAN the time units are generic, they are not related to seconds or minutes. This scheduled time represents the completion time of the activities. Within the scope of these activities are the many instantaneous activities.

Time Duration Activities

Search: This activity is actually a continuous activity modeled as a discrete event. Every soldier performs a search every 20-30 time units. New search cycles are scheduled after unsuccessful searches or after engagements.

Select: This event is scheduled after a successful search cycle, when the observer must assess his potential targets and pick one to engage.

Direct fire: This event is scheduled by the select event. Soldiers must stop, aim, and fire at their targets.

Move: This event determines a new location for the soldier performing the activity and computes the time to start the next move (finish the current move).

React to fire: This event is scheduled by the direct or indirect fire engagement. Targets determine they are being engaged and react in a stochastic manner based on their current posture and move status.

Indirect fire: This event is scheduled by the BLUE squad leader when he detects one or more targets. Time is required to reference the firing data, prepare the rounds and for the rounds' time of flight.

Change direction and formation: This event is scheduled by either squad leader when they select a target to engage that is more than 25 degrees off their direction of travel. The squad leader fires, then decides to direct his squad to alter their direction.

Breach obstacles: When a squad hits a wire obstacle, it takes time to breach it. Squads change their posture and move status until the breach is affected.

The third type of activity consists of tasks required to perform system maintenance activities. These activities are not scheduled, but occur as part of the overall system activities.

System Maintenance Activities

Initialize data sets and event calendar

Select the next scheduled event and delete it from the calendar

Ensure simulation clock does not advance past passage of real time

Update the simulation clock

Add new events to the event calendar

Generate pseudorandom variables

Transfer program control to proper event subprograms

Terminate simulation when terminating conditions are reached

Store historical data for system status reports and final output

Process summarized data for final output

Model Development. Having decided on what objects and entities SPARTAN would have and the activities that these entities would perform, the next step in the model formulation stage was the development of the model itself. This stage had three subprocesses: building a database, developing an event scheduling routine, and adding various routines to perform the modeled activities.

Creating a Database. This was the first step as all objects refer to the database for their attribute values (which, in turn, determine event outcomes). There are five types of default data files, totaling ten files altogether (Table 6). These data files were not all developed up front, several of them were built as new combat processes were added and deterministic methods of deciding outcomes proved too expensive in terms of computation time.

As data files were developed, preprocessors were added to allow users to view and edit some of the data and create their own scenarios. CPT Cox had originally accomplished this using record arrays, but elected to finally use ASCII

Table 6 Default Data Files

FILENAME	DESCRIPTION (size)
mapl.dat	terrain data file (50x50x3)
event.dat	initial event list (99x3)
soldat.dat	soldier attribute list (12x15)
M16.dat	M16 Pr(hit) table (8x3)
AK74.dat	AK74 Pr(hit) table (8x3)
SAW.dat	SAW Pr(hit) table (8x3)
cor.dat	Pr(acquisition) table (10x3)
INFW.dat	Pr(detection) table for targets in wooded areas (10x3)
INFNW.dat	Pr(detection) table for targets not in wooded area (10x3)
THRESHOLD.dat	target-observer threshold detection levels (12x12)

files as record arrays were too difficult to view. I decided to keep the flat file format, despite the greater requirement for storage space. (This caused a memory overflow problem when I attempted to compile SPARTAN, because the data files were so extensive. The solution was to make data arrays within SPARTAN "$DYNAMIC", thus giving them memory addresses outside the 64K set aside by DOS for the executable program.)

Event Set Management. SPARTAN's default time advance mechanism is the hybrid event step process described in Chapter II. User's have the option of making SPARTAN a strictly event step processing model however, through the Alter Terminating Conditions menu. In either process, event set management is the same.

CPT Cox listed event set management as the most difficult task in designing the original SPARTAN (5:51). He went through several versions before deciding on a double linked list approach.

I spent several hours trying to figure out the double linked list approach and why it was used in the original SPARTAN. With only a maximum of 12 entities and five scheduled event types, the event calendar rarely exceeded 30 events and was frequently less than that. Also, users could not see the event manager functioning, so its value as a demonstrator was nil. Consequently, the double linked list method seemed overly complicated for the number of activities that needed to be scheduled and I did not use it.

Instead, I decided to use a simple event array that contained only the scheduled event type, actor, and time. As the current scheduled event is completed, at least one type of time duration activity is scheduled. SPARTAN calls the SCHEDULE subprogram and files the event in the first empty EVENT array row. Program control then passes to the EVENT subprogram. Using the just scheduled activity's start time as a base, EVENT checks all occupied EVENT array rows and identifies the event with the lowest start time. This event is pulled from the EVENT array row (which is then zeroed out) and program control passes to the event subroutine referenced by that event.

This method of event set management must check all events in the EVENT array each time a scheduled event completes. The advantage to my method is that the problem and computational overhead involved in unlinking and relinking lists as events are added and deleted from the event calendar is eliminated.

Model Enrichment. There were actually two separate programs to be built, the preprocessor and the simulation. Although developed separately, the process of building both was cyclic, involving building and testing code for a module, adding it to the main program, refining it, and going back to edit and debug previously written modules as errors became apparent with new model tasks.

The first part of the model to be developed was the initialization subroutine. This subprogram read the data files and created the data arrays for the rest of the program. To avoid the problems that CPT Cox had with data integrity (5:54), I elected to keep all data values as single precision values instead of a real/integer mix. This caused no problems, but I had to be careful in the choice of my values for comparisons for IF-THEN statements.

Because SPARTAN was designed as an instructional tool, I felt that the graphical representation of the battlefield was a very important part of the model. Consequently, the map screen was the first module to be developed after the data initialization module. Combat processes were then added to the model in the order required to test their performance. The first combat module added was the movement module. This was followed in order by the search, line of sight, select, direct fire engagement, impact assessment, react to fire, obstacle breach, and adjust direction and formation modules. In most cases, new modules caused additions and revisions to previously added modules as system activities became more and more complicated. Debugging was aided by keeping code referencing future model subprograms out of the current version of the model. In addition, QuickBASIC has an excellent DEBUG feature and a line editor that checks code as it is entered.

A typical improvement cycle involved creating a subprogram and testing it in isolation. To do this, I wrote a partial SPARTAN program to test separate subroutines. Next, I added the subprogram to the main program. I also added code that caused written values and messages to appear on the screen as different decisions were made in the new subprogram. Revisions were made as necessary. No more subprograms were added until the current version of the model was functioning as intended.

Randomness. Because SPARTAN is a replication of combat, chance outcomes play a major role in simulation logic. Simplicity rather than accuracy, however, was the goal in dealing with probability distributions in SPARTAN. Consequently, SPARTAN uses the QuickBASIC Uniform (0,1) pseudorandom number generator as the basis for determining outcomes for all stochastic processes.

The most frequently used probability distribution was the Uniform (0,1) distribution. SPARTAN also employs the triangular distribution function written by CPT Cox (5:56):

> This distribution was chosen because the transform operation is efficient and the output can be used to represent both skewed and symmetric distributions. The function is given a low, high, and mode values and returns a value within this range. The algorithm for

the transform was adapted from Pritsker (19:713). This transform provides a rough approximation to a normal distribution when the mode is centered and the extreme values are assumed to be within two standard deviations from the mean.

Although some event outcomes are derived using deterministic methods, most are the results of a Bernoulli trial. Probability of hit tables, probability of acquisition tables, and probability of detection (given targets are acquirable) tables were all computed beforehand. These tables are referenced by the appropriate subprogram and compared to the random variable. This method is similar to ones used by both JANUS and CASTFOREM for various weapons' data. In the case of SPARTAN, these probability tables are based in large part on actual accuracy data from AMSAA and on NVEOL data. Some interpolation and approximation was necessary, but overall the results are close to the data used in more capable models. These tables were originally created using MATHCAD templates, examples of which can be seen in Appendices A-C.

Instructional Components

Since SPARTAN was designed as an instructional tool, it has several features not normally found in purely analytical models. Some of these features can be found in the preprocessor, which not only allows the user to alter default data or create his own files, but is also designed as an instructional device. Another feature, found in both the preprocessor and the simulation, is the help file. The third feature is the graphics of the simulation. Simulation output also is an instructional component.

<u>The Preprocessor</u>. SPARTAN's preprocessor, STARTUP.exe, was created to allow users to load, view, or edit default data files and to assist them in learning about what their options are for altering default data. Basically, STARTUP.exe reads the .dat extension data files into arrays, allows the user to edit them, and creates .exp files for SPARTAN to read.

STARTUP is menu driven and, like SPARTAN, has extensive help files. The four data files that STARTUP allows the user to edit are the terrain data file, the soldier attribute list, the initial event list, and the probability of hit tables.

<u>Terrain Editor</u>. Unlike the first version of SPARTAN, the terrain attribute list is hard wired. Because the data must match the graphical representation on the screen for the screen to be meaningful, the data files are closed to users. The terrain editor, however, is designed to assist first time land combat modelers. It offers the options below:

1) View map--allows the user to view the map.

2) Add obstacles--allows the user to add one wire obstacle. Creates OBS file for SPARTAN.

3) View terrain dat--allows the user to view terrain cell data.

4) View elevation data--allows the user to see how terrain cell elevation data dictates contour lines.

5) Line of Sight--allows the user to pick observer location and checks the line of sight for user input ranges and fields of view.

The terrain data file is call MAP1.dat and contains 3 attributes for each of 2500 terrain cells as discussed earlier.

Soldier Attribute Editor. The Soldier Attribute Editor allows the user to accomplish tasks listed below:

1) View BLUE soldier attributes--allows the user to view selected BLUE soldier attributes.

2) View RED soldier attributes.

3) Add soldiers--allows the user to add soldiers (for a maximum of 12 soldiers).

4) Delete soldiers.

5) Edit soldier attributes--allows the user to edit a selected soldier's attributes.

6) Pick formation and location--allows to pick the BLUE squad leader's location and one of four BLUE formations. Automatically updates position data for the remainder of the squad.

Probability of Hit Editor. The Probability of Hit Editor allows the user to either review current Phit tables or to create his own Phit tables. SPARTAN uses the Polya-Williams approximation (10,13) to compute the single round hit probability of a hit on a rectangular target. Then, assuming each round within the burst is independent, computes the probability of a least one hit for each burst of fire.

Polya-Williams needs both vertical and horizontal aim and ballistic error. STARTUP will show the raw error data and the computed Phit for each weapon referenced by target posture and range.

Users can also input their own weapons' data, but they need aim and ballistic error (vertical and horizontal) for ranges of 100 to 800 meters (in 100 meter increments). STARTUP will then compute and show the Phit tables. For the table to be used however, the user must alter at least one soldier's weapon to reflect the new weapon type.

<u>Event List Editor</u>. The Event List Editor allows the user to accomplish the tasks listed below:

1) View initial event list.
2) Add events to initial list.
3) Delete events from initial event list.

The event list has an event type, time, and actor for all scheduled events.

<u>Help Menu</u>. In addition to the above options, STARTUP offers help on all four subjects. Users can access the help menu from the main menu or from the editor in which they are currently located.

<u>SPARTAN</u>. SPARTAN has three main instructional components: the help menu, the graphical representation of simulation events, and the battle statistics offered as output during simulation runs or after terminating conditions are met.

<u>Help Menu</u>. The help menu in SPARTAN can be accessed from the main menu. The help menu contains files on each of nine combat processes: search, select, move,

direct fire engagement, indirect fire engagement, react to fire, impact assessment, terrain representation and obstacles, change formations, and general information about setting up and running SPARTAN. These are usually several screen pages in length and go into detail about simulation logic and equations.

Simulation Graphics. The simulation graphics of the current version of SPARTAN differ greatly from the original version's: the terrain looks like a military 1:50,000 scaled map, soldiers are stick figure icons of the appropriate color, firing is represented by a line clearly being drawn from the observer to the target accompanied by a burst of sound representing the number of rounds being fired, indirect fire is represented by a line drawn from the firer to the impact point and an explosion scaled to the bursting radius of the round, and icons change colors when killed (as they did in the original). All of these features are designed to aid users in understanding what each icon is doing and thus understand why simulation results are the way they are.

In addition to graphics, SPARTAN shows messages on the top of the screen to reflect both simulation activities and player (icon) communication. The current simulation clock time is shown in the upper right corner, as are the current event and actor. Messages reflecting detection and squad leader commands to his squad are shown in the upper left corner, as are the results of engagements and the probability of hit that caused the outcome. These messages too, help the user to understand model workings.

Output. SPARTAN offers output both during the simulation run and after simulation termination. During simulation runs, users have access to the following information:

1) Current soldier attributes. Translates current soldier attribute values into English for both BLUE and RED.

2) Current potential target list. Shows the current values in the potential target list. Allows the user to assess who has acquired who, and who has tried and failed to acquire who.

3) Current Event list. Shows the current event list (in order of execution) in English.

4) Current battle statistics. Shows the current WIA and KIA status for both sides.

After the simulation terminates, the user also has the option of reviewing the above tables and a Kill Card which reflects how many soldiers were hit by weapon type and the maximum, minimum and mean range of hits for each weapon. SPARTAN also creates a history file. This file contains every scheduled event type, the time it occurred, and the actor who performed the event.

Model Assessment

Assessment of the new SPARTAN was much like that of the original version. Primarily it revolved around an assessment of how well the original project goals were met and how well it met evaluation criteria established by the 1979 GAO report on Defense Analysis.

Assessment Process. The assessment process for SPARTAN took place in three phases. The first phase was an evaluation by me. The thesis advisor then evaluated the model and provided more guidance and suggestions as to focus of effort and improvements. The final assessment was the same as used by CPT Cox, blind testing.

Blind Testing is suggested by James Dunnigan in his book, The Complete Wargames Handbook, and involves a series of laboratory tests performed by personnel with backgrounds similar to those of the targeted audience. For SPARTAN, I used two iterations of blind testing. In the first test, three students of a land combat modeling course were issued a copy of the model, supporting data files, and a user's guide. Two students were Army officers with extensive experience in small infantry unit operations. The third student was an Air Force officer. The second group consisted of three Army officers of mixed background experiences. Students were given no verbal instructions, other than to comment on problems with using the preprocessor, understanding the user manual, or perceived problems with model operations. Improvements were made both to the model and to its operating instructions after each iteration of testing.

GAO Criteria Assessment. The GAO report lists five criteria for use in evaluating models. These are: documentation, validity, verification, maintenance, and usability (11:9). These are discussed below.

Documentation. Documentation refers to the documents and comments that accompany the model (12:27-28). The purpose of documentation is to allow someone other than the model builder to understand the model assumptions, methods, and inner model workings. In this sense, SPARTAN has two types of documentation. The first is the user's manual and the help files in the preprocessor and the simulation. This documentation is intended for the user who does not wish to get into the nuts and bolts of programming. The second type of documentation is the thesis. This documentation is intended for the user who wishes to go into great detail about algorithms and equations.

Most user's of SPARTAN will use the user's manual and the online help as opposed to the thesis. The user's manual and online help were designed to be used in concert with each other, neither are stand alone documents. Using CPT Cox's findings, I elected to make the user's manual more of a general document about how to use SPARTAN and the preprocessor. It has narrative outlines about model functions and methodology. The online help contains much more detail about algorithms and equations used in specific processes.

The thesis document was intended to provide a more thorough discussion about model formulation (including assumptions, goals, and limitations) and the specific techniques used to model various combat processes. This chapter accomplishes the former while Chapter IV discusses in detail how modeling combat processes was accomplished. Appendices A-C contain the templates used to develop the probability tables and Appendices E and F contain the program code for the preprocessor and SPARTAN. All this information should be enough to allow a user, familiar with QuickBASIC, to begin altering code after a few weeks study and should satisfy the GAO criteria.

Validity. CPT Cox listed three types of validity from the GAO report (5:64). These were data validity, theoretical validity, and operational validity. This version, like the original version of SPARTAN, makes no pretenses of being a "valid" recreation of a small unit combat. The objective was to demonstrate current model techniques in the present generation of high resolution land combat models. Nevertheless, I strove for as much data validity as possible. Weapons' data, for example, is the same accuracy data used by JANUS and CASTFOREM. On the other hand, acquisition data is simulated, although it is close to data used by "real" models. Therefore, SPARTAN has only partial data validity.

There is also partial theoretical validity in SPARTAN, as most of the algorithms are adopted from JANUS and CASTFOREM documentation. However, even these algorithms are suspect. A search of JANUS and CASTFOREM documentation reveals many heuristics and data used for purposes far outside its original test conditions (the NVEOL applications are a good example of this).

Operational validity does not apply to SPARTAN either, as it is contingent on theoretical and data validity.

In SPARTAN's case, validity is the ability of SPARTAN to meet its intended goal of providing a useful tool for beginning land combat modelers. Blind testing revealed that this goal could be met.

Verification. Verification ensures that computer code is performing its functions as intended. Verification of code began with initial code testing before modules were added to the main program. A special "little" SPARTAN was developed for testing subprograms. Then, after these subprograms were added to the main program, parameters were altered to simulate a wide variety of conditions to ensure all parts of the program were functioning correctly. In many cases I accepted limits that CPT Cox had established. Blind testing turned a few verification problems, as outsiders applied previously unused scenarios to the simulation. Most of these problems were quickly corrected.

Maintenance and usability are discussed in the context of meeting design objectives below.

Model Objective Assessment. There were four objectives in designing SPARTAN: portability, usability, simplicity, and applicability. All of blind Testing group felt that the goals were met.

Portability. SPARTAN was designed to run on most IBMs or IBM clones. It was tested on computers ranging from the 286 it was designed on, to the 486 at the AFIT computer laboratory. It runs much faster on 386s and 486s of course. One result of this finding was the option of relating the advancement of the simulation clock to the passage of real time. This was necessary in order to slow simulation runs and allow users to see and understand model functions.

Usability. The objective of usability benefitted the most from the blind testing. Having fresh eyes and brains run the simulation really aided in making the model user friendly. The preprocessor was modified greatly after the first round of testing. All testers felt that the simulation was usable and that modelers would benefit from the simulation.

Simplicity. Simplicity benefitted from the structured programming that QuickBASIC allows. New processes need only be written, then plugged in (or yanked) as a module or subprogram of the main program in QuickBASIC. Because SPARTAN is menu driven and user input is limited, the program is very simple to operate, this also relates to the previous objective. This objective also helps meet the GAO criteria of maintenance, since modular coding, the programming language, and simple algorithms make SPARTAN easy to maintain.

Applicability. This objective goes back to the validity issue. All processes were modeled on current techniques and all data is real or as real as possible. Testers felt that the processes were applicable and that beginning modelers would learn by using SPARTAN.

Summary

Designing SPARTAN was an iterative process of model formulation, coding, implementation, debugging, and modification. This development process was modeled after the processes recommended by Pritsker and the conical methodology. Using these processes, the following model components were developed:

1) Input Data Files

2) STARTUP preprocessor

3) SPARTAN simulation with subprograms for:
   - scheduling events
   - initialization of data bases
   - transferring program control to sub programs
   - terrain representation
   - target acquisition
   - line of sight determination
   - target selection
   - direct fire engagements
   - indirect fire engagements
   - round impact assessment
   - reaction to fire
   - command and control
   - online help
   - battle statistics
   - movement
   - obstacle breaching

4) Model documentation

5) User's manual

The next chapter reviews how specific combat processes were modeled.

IV. Combat Processes

Introduction

This chapter focuses on the algorithms and equations used in SPARTAN to replicate specific combat processes. All methods of modeling combat processes are based on techniques used in the present generation of high resolution land combat models. For each process, there will be a description of the technique (including formulas and algorithms), a discussion of the limitations and assumptions involved in the method, and a flowchart demonstrating the algorithm. The processes are discussed in the general subject order of move, search, shoot, and react for ease of discussion of process flow.

Movement

Like JANUS and CASTFOREM, this version (as well as the original version) of SPARTAN models movement, a continuous process, in discrete steps. Both versions of SPARTAN also resemble JANUS and CASTFOREM in that movement is a function of the moving entity's and the terrain cell's attributes. This SPARTAN differs from the original version in that it models obstacles and moves entities in a manner that maintains unit formations.

<u>Step Size</u>. SPARTAN advances all entities 20 meters per move, regardless of entity posture. The new entity location is computed every move cycle based on the two equations below (10:40-15):

$$Xnew=Xold+20\times COS(DIR) \tag{1}$$

$$Ynew=Yold+20\times SIN(DIR) \tag{2}$$

The Xold and Yold represent the current horizontal and vertical coordinates of each entity and DIR represents the direction of movement of that soldier. All three are soldier attributes. Unlike JANUS and CASTFOREM, in which entities change direction according to preset movement control points, the direction of each soldier remains constant unless changed by the squad leader. More will be discussed about this subject in the section about the CHANGE DIRECTION AND FORMATION subroutine.

<u>Movement Time</u>. Since the movement distance is constant for every move, movement time is used to reflect variability in movement rates. In SPARTAN, this time is a function of the posture of the soldier, the mobility factor of the terrain cell in which he starts his move, and a pseudorandom Uniform (0,1) variable

$$movetime=10+10\times\frac{RND}{mobilityfactor\times posture} \tag{12}$$

Since mobility factors range from .1-1, with lower values having worse mobility and posture values range from .25-1, again with reduced postures having lower values, the effect is to increase travel time for the 20 meter increment for decreased trafficability or reduced posture.

All movement times are decided by the squad leader for either side. The designated squad leader is the first entity scheduled for a move and his attributes dictate the move times for the remainder of his squad. This was done to maintain formation integrity for both sides. This is a realistic limitation, as the first command of the squad leader to his squad is to "follow me and do as I do".

Neither roads nor the slope of the terrain have an impact on soldier movement times. This is not unrealistic, as dismounted infantry moving in this type of scenario would not be affected by slope or the presence or absence of roads. Obstacles do affect movement.

Obstacles. Obstacles are input by the user in the preprocessor's terrain editor. The only obstacle type that SPARTAN recognizes is a wire obstacle (triple strand concertina). Unlike JANUS and CASTFOREM, the only option for soldiers who encounter obstacles intersecting their path of travel is to breach the obstacle. Also unlike JANUS, there is no such thing as a "friendly obstacle", obstacles impede both sides, not just the enemy.

Because the movement time is determined by the mobility factor of the terrain cell in which the move starts, soldiers do not realize that they are breaching until their new location is computed. When a soldier detects that he is in a terrain cell with an obstacle, the entire squad changes into a prone-nonmoving status to begin breaching. In addition, all moves for that side are deleted from the event list and an ENDBREACH event is scheduled for 100 time units in the future. At that time, the squad changes back to a standing and moving status and moves for each squad member are scheduled.

SPARTAN Movement Process. In SPARTAN, movement is controlled by one subprogram called MOVE (Figure 3). This routine accomplishes a number of actions including determining movement times, updating soldier attributes, and updating graphics.

The first function MOVE accomplishes when called, is to check the status of the soldier to ensure that he can move. If the soldier is in a nonmoving status, dead, or his side is in a breach mode, the subprogram ends and program control passes back to the main program to allow sequencing of the next scheduled event. Some of these checks are redundant (as dead soldiers should be in a nonmoving status), but were left in the code to ensure that there was no programming error.

MOVE then checks the soldier's attributes to ascertain his screen location. If the soldier is on the screen, his old position is erased. After computing the new location based on the soldier's direction of travel, the new position

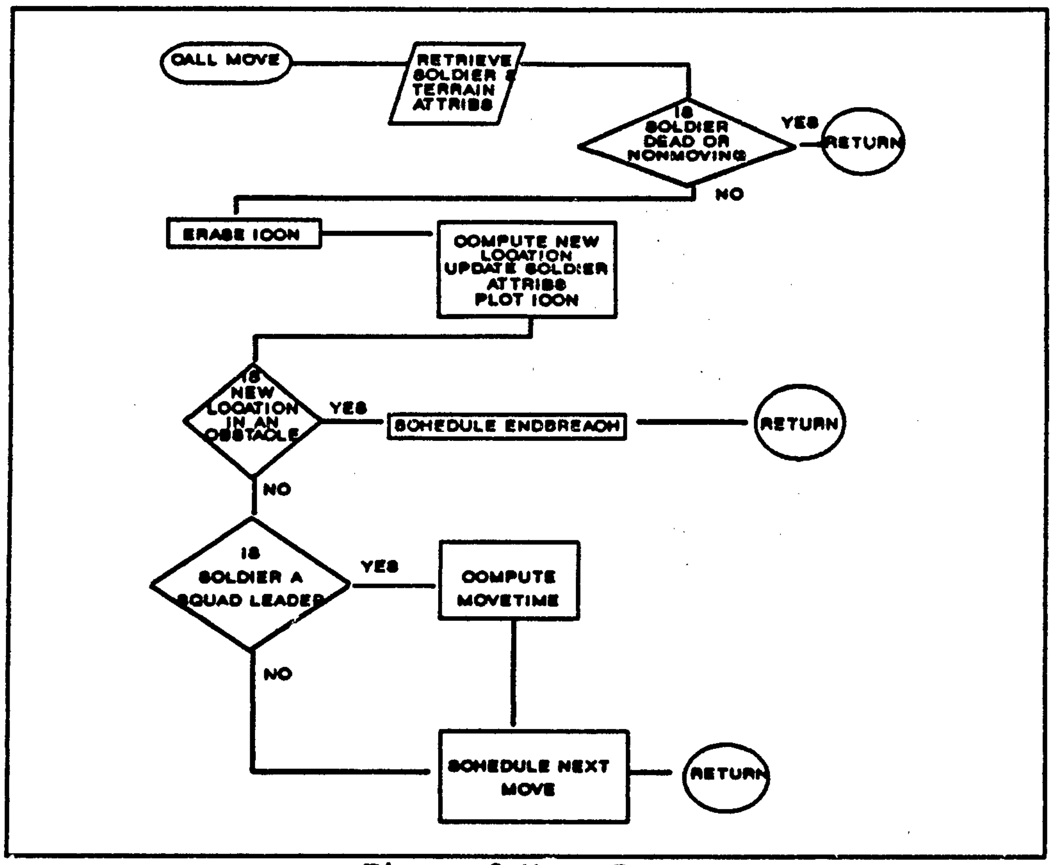

CALL MOVE
RETRIEVE SOLDIER & TERRAIN ATTRIBS
IS SOLDIER DEAD OR NONMOVING
YES
RETURN
NO
ERASE ICON
COMPUTE NEW LOCATION UPDATE SOLDIER ATTRIBS PLOT ICON
IS NEW LOCATION IN AN OBSTACLE
YES
SCHEDULE ENDBREACH
RETURN
NO
IS SOLDIER A SQUAD LEADER
YES
COMPUTE MOVETIME
NO
SCHEDULE NEXT MOVE
RETURN

Figure 3 Move Process

is plotted on the screen. This location represents where the soldier will be at the end of the move time (beginning of the next move event). MOVE then checks the terrain cell attributes of the new location and, if there is an obstacle present and the squad is not breaching, a BREACH is called, all squad movement stops, and program control passes to the next calendar event.

MOVE next checks to see if the soldier is a squad leader. If he is, an outline is drawn around the icon (to designate him as squad leader) and the move time for that side is computed and scheduled for the squad leader.

Program control then passes to the next calendar event. If the soldier is not a squad leader, his next move event is scheduled based on his squad leader's previously determined move time.

Limitations and Assumptions. The low level of detail in this movement process results in the following limitations and assumptions:

1) Entities are in an iterative process of moving and stopping rather than continuous movement.

2) Soldiers always move in 20 meter increments, regardless of the situation. In reality, there might be cases where moves less than 20 meters might be desirable.

3) Slope does not affect movement speed.

4) Trafficability is determined by the starting terrain cell and is assumed to remain constant throughout the move.

5) Because the battle duration is short, soldier fatigue is not a factor.

6) Movement times are scaled to look "right" and do not reflect real times.

7) Computations are scaled for graphics.

8) If a soldier changes posture during movement, that change does not affect the move in progress.

9) Soldiers cannot read terrain, and do not alter movement based on obstacles, avenues of approach, or cover and concealment.

Target Search

SPARTAN uses a search process based on the Night Vision Elector-Optical Laboratory (NVEOL) model. Unlike the original SPARTAN, this version of SPARTAN more closely replicates the process used by JANUS and CASTFOREM. Like movement, search is a continuous process modeled in discrete steps.

Search is actually a three phased sequential process. If the success threshold at any phase is not reached, then that search sequence is unsuccessful and target detection is not achieved. The three conditions for target detection are:

1) The target must give off sufficient signature to be detected by the observer.

2) The observer must have line of sight to the target.

3) The observer must be looking at the target.

This version of SPARTAN, like the original, assumes a full 360 degree search pattern. JANUS also does this, but changes the field of search to 180 degrees when entities are stopped, while CASTFOREM's search pattern is a function of time. While a 360 degree search pattern replicates a squad maintaining good security, it eliminates the effects of focusing on sectors of responsibility or known enemy positions and probably does not accurately reflect normal soldiers' search patterns.

This version of SPARTAN differs from the original in that the search process is not memoryless. In the old SPARTAN, every search process was new, soldiers did not remember detecting any enemy on previous searches. Also, a squad member's successful enemy detection did not improve the probability of other squad members detecting that enemy soldier. This version of SPARTAN attempts to correct this.

Condition I: Sufficient Signature. The first condition that must be met for successful target detection is that the target gives off sufficient signature for the observer to detect it. Like JANUS and CASTFOREM, SPARTAN assigns every possible observer-target pair a random threshold level of resolvable cycles (see Chapter II). Each of these threshold levels is from a random lognormal distribution with the underlying normal distribution having a mean of 3.5 and a standard deviation of .698 (17:25-32). These random threshold levels of resolvable cycles are then translated into threshold levels of detection. This has the effect of injecting randomness into the detection algorithm and can be justified by the fact that some observers will discern targets more quickly than other observers (and by the same thought, some targets are more discernable than others). The observer's ability to discern a target given unlimited time observing his sector determines whether or not he exceeds this threshold level. The computations for this threshold data are Appendix A.

The observer's ability to discern a target, called Pinf, is a function of target-background contrast, observer-target range, sky-to-ground brightness ratio (SOG), and a atmospheric attenuation coefficient. The first step in computing Pinf is to compute the attenuated contrast, which is the apparent target-background contrast taking in the above factors. This is computed by Equation (4) from Chapter II(14:25).

$$Attenuatedcontrast=\frac{targetcontrast}{1+SOG\times(e^{atmosatten\times range}-1)} \quad (4)$$

SOG is set at 2.5 (14:25) and the target contrast for the visible spectrum is between .2 and .3. I used .3 for nonwooded areas and 2.9 for wooded areas. The documentation for JANUS(L) did not give atmospheric attenuation coefficients, so I experimented with various values and settled on using .01. The results using these coefficients seemed reasonable.

Once the attenuated contrast was computed, the next step was to compute the target dimension in milliradians

$$milliradians=\frac{MinTgtDim(meters)}{Range(km)} \quad (13)$$

Where the minimum target dimension was determined by target posture (Table 7).

Table 7 Target Dimensions

POSTURE	DIMENSIONS	MINIMUM DIMENSION
Standing	1.8m x .8m	.8m
Crouching	.9m x .8m	.8m
Prone	.45m x .8m	.45m

It was then necessary to determine the cycles per radian, which is a function of a sixth degree polynomial of the natural log of the attenuated contrast. Having determined the cycles per radian and the radian per target dimension, it was then possible to determine the cycles resolvable for the observer (cor) for different ranges and target postures. These values were entered into tables by target posture and range (in 100 meter increments) and for the target background contrast (depending on whether the target was in a wooded area or not). Calculations for these tables are in Appendix B.

Once the cycles resolvable for the observer (cor) was determined, the next step was to determine the Pinf value for target posture, range, and background contrast. This was done using Equation (5) from Chapter II (14:26).

$$Pinf=\frac{\frac{cor}{3.5}^{2.7+.7\times\frac{cor}{3.5}}}{1+\frac{cor}{3.5}^{2.7+.7\times\frac{cor}{3.5}}} \qquad (5)$$

Where "cor" is the cycles resolvable by the observer and 3.5 is the average cycles resolvable required for target identification. The calculations for this table are also in Appendix B.

If the Pinf in the lookup table for the given range, target posture, and background is greater than the threshold level for the particular observer-target pair, then the target is giving off sufficient signature and the search process tests the second condition for target detection.

<u>Condition II: Line of Sight</u>. If the target is giving off sufficient signature, then it must be determined if the view of the target is blocked by intervening terrain or vegetation. In the original version of SPARTAN, the intervening terrain had no vegetation, only terrain elevation could block line of sight. Also, SPARTAN only

checked LOS every ten meters. It was possible for an intervening terrain cell to block LOS and not be checked if the observer-target line cut across the terrain cell at some angle other than a perpendicular one.

In this version of SPARTAN, LOS is checked in every intervening terrain cell between observer and target. First, SPARTAN determines the observer elevation based on his posture and the elevation of the terrain cell in which he is located. SPARTAN determines the target elevation based on the same attributes. SPARTAN then determines the equation of the observer-target line and the slope of that line based on these two elevations. Then, using the slope of the line and the distance along this line from the observer, SPARTAN compares the elevation of every intervening terrain cell to see if it blocks the observer-target line. If the terrain blocks the line, LOS does not exist.

SPARTAN also takes into account the vegetation of the terrain cells in which the observer and target are standing as well as the vegetation in all intervening terrain cells. SPARTAN first checks to see if the observer is in a wooded area (denoted by a mobility index less than one). If so, it ignores vegetation effects until the observer-target line gets to a clear (non-wooded) area. Once the line enters a clear area, any subsequent vegetation height is added to the elevation of the terrain cell which contains it. This is done using Equation (14). Depending on the mobility factor,

$$elevation = elevation + \frac{10}{mobilityfactor} \qquad (14)$$

this might add 14-16 meters to the terrain cell's elevation. This has the effect of allowing LOS to targets in the same wooded area the observer is in, but blocking LOS to targets in other wooded areas.

Condition III: Detection. If the target is giving off sufficient signature and the observer has LOS, then it must be determined if the observer detects the target. Because of the randomness involved in this condition, SPARTAN directly computes its existence.

First, SPARTAN computes the probability of the target being in the observer's field of view during this search cycle (14:28-29).

$$Pfov = 1 - e^{-\frac{1}{6.8} \times cor \times (1.7 + 3.4 \times RND)} \qquad (15)$$

Were cor is the cycles resolvable by the observer determined in Condition I and RND is a random Uniform (0,1) variable.

Then, using Pinf and Pfov, the probability of detection is computed using Equation (7) from Chapter II (14:26).

$$P(detection) = Pinf \times Pfov \qquad (7)$$

SPARTAN then performs a Bernoulli trial to see if the target was detected.

SPARTAN Search Process. In SPARTAN, the search process is controlled through two subprograms called LOS and ACQUIRE. ACQUIRE is a scheduled event and determines if target signature is sufficient, calls to LOS to determine if line of sight exists, and then determines if detection is made.

When ACQUIRE is called, it first checks to ensure that the observer is alive. If the observer is dead, then control passes to the next scheduled event.

Next, the subprogram checks to see if the observer is a squad leader. If he is a squad leader and wire obstacles are present, a call is made to the subprogram WIRE. WIRE first checks to see if the current direction of travel intersects a wire obstacle. If it does, then LOS to the obstacle is checked and if it exists, a message reflecting obstacle detection is shown on the screen. No other action results from obstacle detection because no direction changes are permitted. If the obstacle does not cross the squad leader's path or if there is no LOS, no detection occurs. After WIRE completes execution, control passes back to ACQUIRE.

When the squad leader finishes obstacle detection or if the observer is not a squad leader, SPARTAN checks all enemy soldiers to see if the observer can detect them. First, SPARTAN accesses the Pinf and COR tables by target range, posture and background. It then compares the Pinf to the observer-target threshold level (Figure 4). If the Pinf is too low, SPARTAN checks the next enemy soldier.

If Pinf is sufficient, ACQUIRE calls LOS to determine if line of sight exists (Figure 5). If los does not exist,

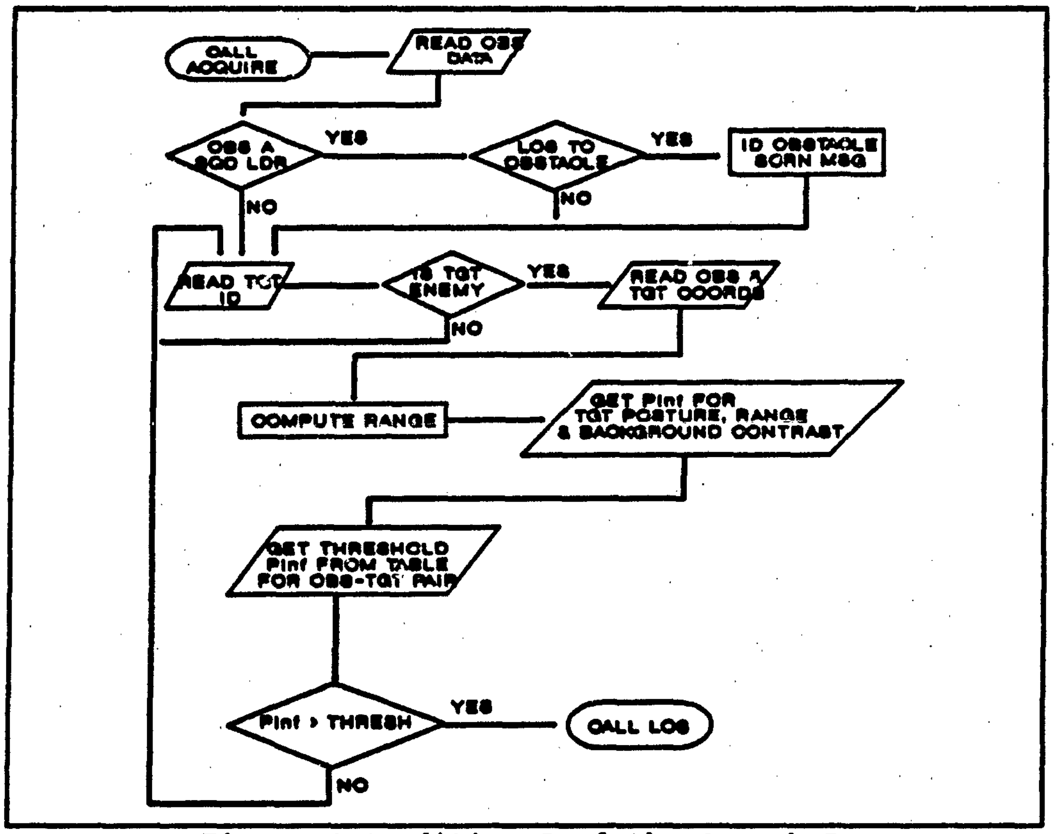


Figure 4 Condition I of the Search Process

SPARTAN begins acquisition checks on the next enemy soldier.

If LOS and Pinf conditions are met, SPARTAN checks Condition III (Figure 6) to determine if detection was successful. If detection was successful, the potential target list for the observer is updated with the probability of detection for that target, a message reflecting successful detection is shown on the screen, a SELECT target event is scheduled (five time units from the current time), and the observer continues to search until he has attempted to detect every enemy target. If the detection was unsuccessful, the search continues until an attempt has been made to detect all enemy soldiers and then a new search

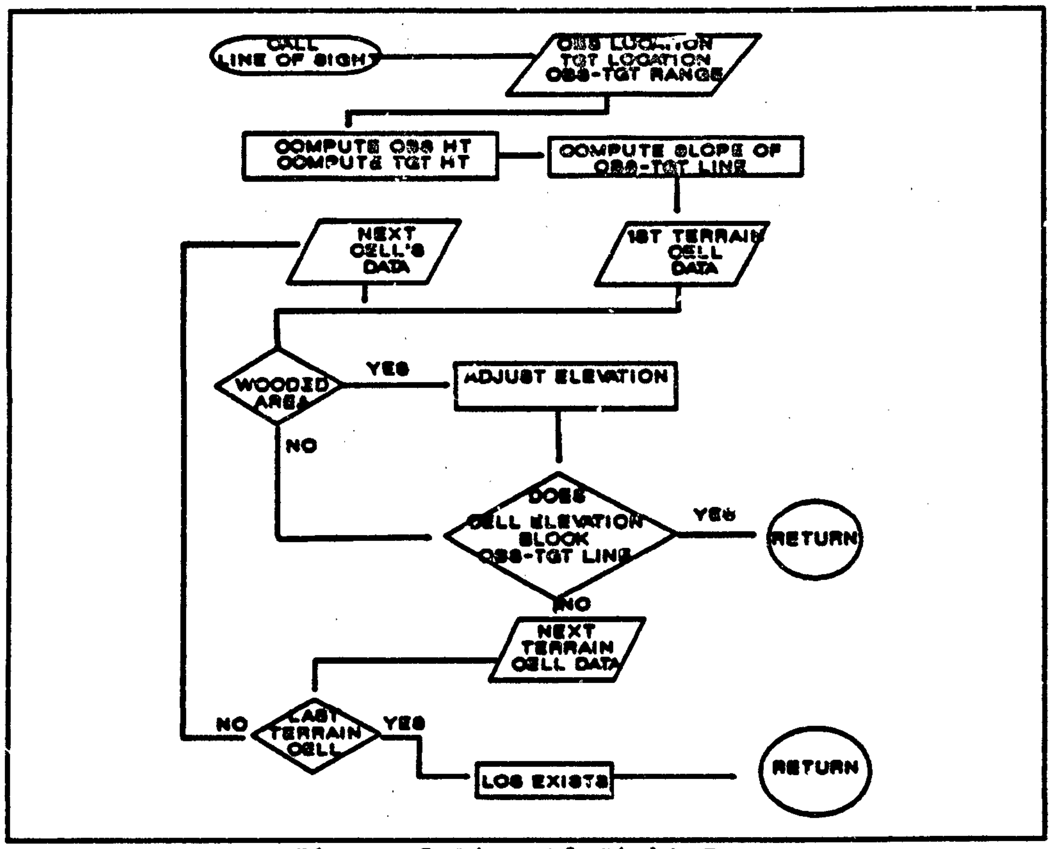


Figure 5 Line Of Sight Process

scheduled. The new search time is random time between 0 and 40 time units in the future.

<u>Heuristics</u>. One heuristic this version of SPARTAN uses is the modification of the target dimension (and thus the "cor") based on target's move and firing status. If the target is moving or has fired in the last 20 time units, SPARTAN doubles the target dimension, increasing Pfov and the probability of detection. This heuristic is the same used by JANUS (Chapter II) and has the effect of focusing the soldier on a target area because of some target action.

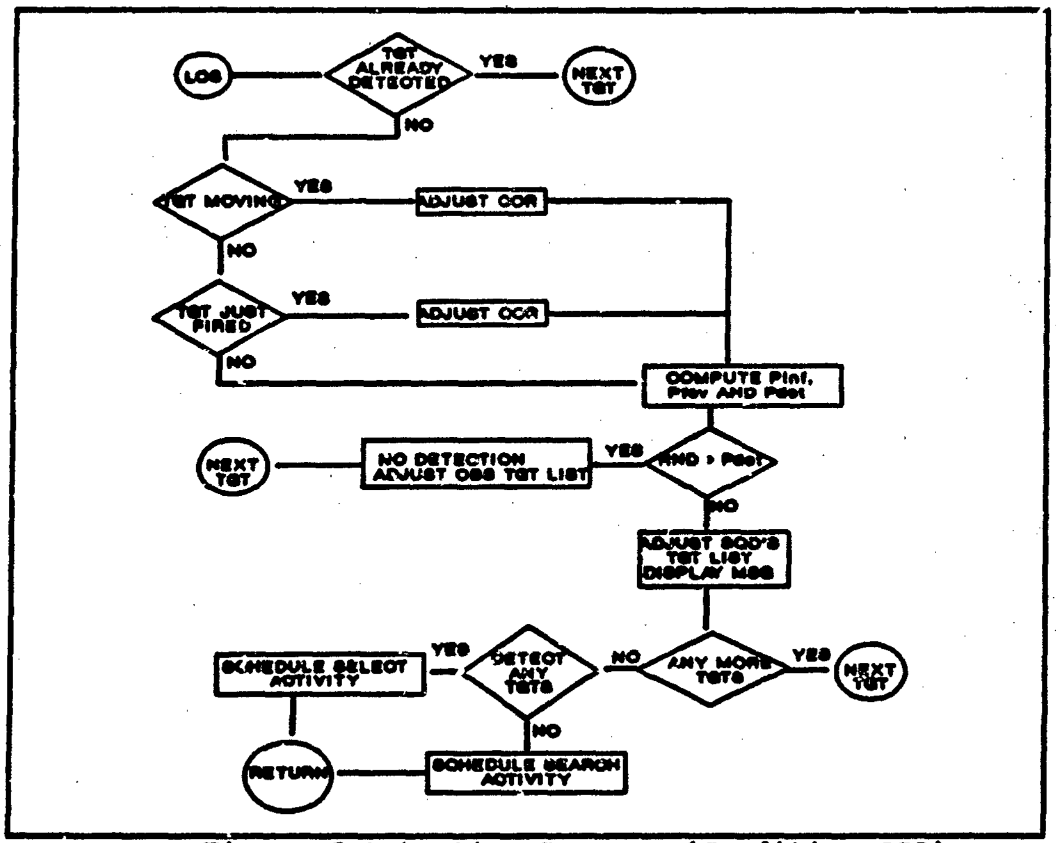


Figure 6 Detection Process (Condition III)

Another heuristic, developed for SPARTAN to account for squad communication in the event of a successful detection, is the modification of the potential target list. In this heuristic, the probability of detection is multiplied by a negative linear modifier (-.4) and put on the potential target list for all soldiers on the observer's side. The modifier is negative to reflect that detection has not occurred for that observer target pair.

A third heuristic is employed in the event of a unsuccessful detection (given Pinf and LOS exist). In this event, the probability of detection is modified by (-.5) and added to the observer's potential target list.

The reason for this is to make the search effects cumulative.

The results of these last two heuristics is that in subsequent searches, if the observer's potential target list value for a particular target is negative, his probability of detection for a given target is increased because the absolute value of the negative Pdet is added to the computed probability of detection before performing the Bernoulli trial. Thus, if a fellow squad member detected a target or if an unsuccessful search was conducted, the chances of detection on subsequent searches is increased.

In addition to modifying the potential target list values based on previous searches, the potential target list value for a target is returned to zero if line of sight ever is lost, either on a subsequent search or in a direct fire engagement.

Limitations and Assumptions. The following limitations and assumptions exist in this search process:

1) Because of the many steps involved in computing Pinf and cor, values are in table form. Interpolation must be used for ranges not in 100 meter multiples.

2) Sky-ground brightness ratio, atmospheric attenuation factor, and background contrast ratio are constants.

3) There is no partial line of sight. If the observer can see the target's eye, then SPARTAN assumes the entire target can be seen.

4) Elevation throughout the entire terrain cell is constant. It is possible for a target to be next to a neighboring terrain cell and not be seen

because of the elevation difference of the two cells.

5) Linear modifiers used in heuristics for successful and unsuccessful detections are arbitrary. They seem reasonable based on experience, but no data of any kind was involved in deriving them.

6) The heuristics used for targets that are shooting or moving are those used in JANUS (14:2-32), but no data is presented in the JANUS documentation to support them.

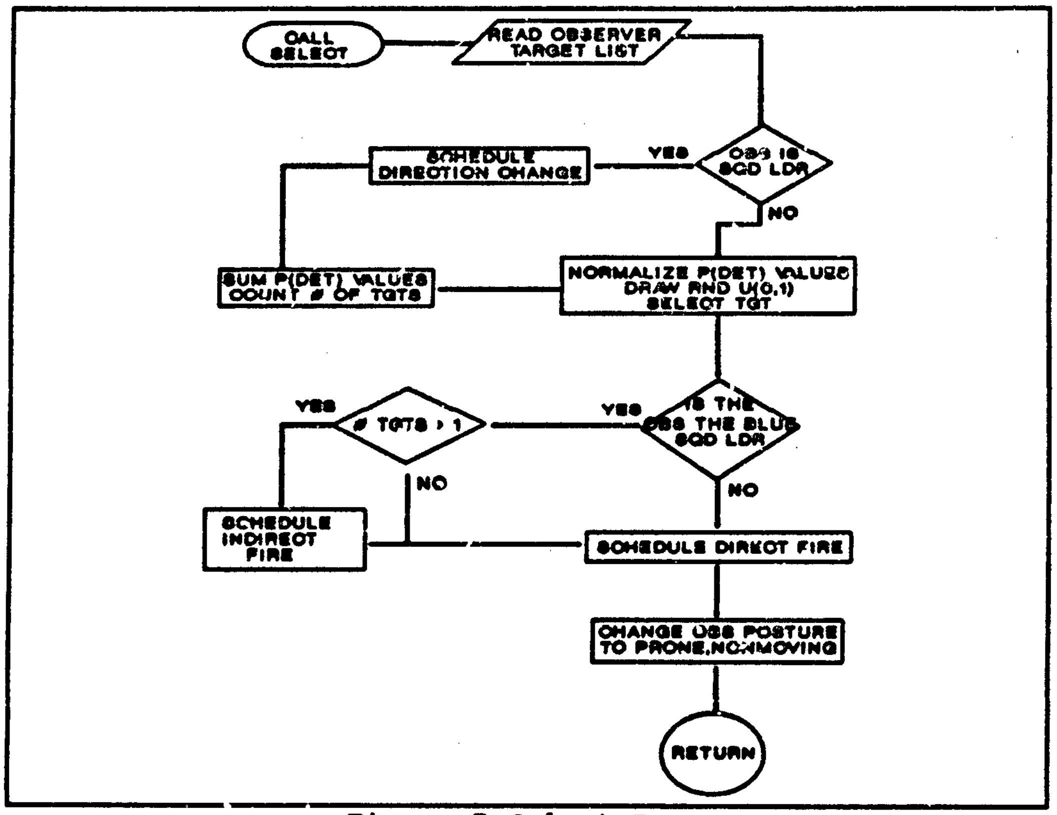
7) Targets are always perpendicular to the observer. Target dimensions are not adjusted for side or off-center view points.

Target Selection

Once an observer has detected one or more targets, he must decide if he wants to shoot at a target, which target he wishes to shoot, and which of his weapons he wishes to use. Target selection is the process that decides all of this. The SPARTAN selection process, SELECT, is more like the process employed by JANUS which uses range, weapon accuracy, and probability of hit to decide which target to engage, rather then the process employed by CASTFOREM, which uses weighted decision variables.

This version of SPARTAN is like the original version in that the probability of detection value is used to determine target selection. This is because the probability of detection takes into account range to target, target dimensions, and signature of target (target movement or firing). Thus targets that are close, have fired, are standing, or are moving will have larger probabilities of detection than prone nonmoving targets. This corresponds to a target sel   on process wherein observers engage the target they perceive is most dangerous.

SPARTAN Target Selection Process. The subprogram SELECT (Figure 7) accomplishes several tasks in SPARTAN. It selects a target for engagement, schedules the engagement, and, if the observer is a squad leader, may schedule a formation and direction change. SELECT also schedules indirect fire missions.


CALL SELECT
READ OBSERVER TARGET LIST
OBS IS SQD LDR
YES
NO
SCHEDULE DIRECTION CHANGE
SUM P(DET) VALUES COUNT # OF TGTS
NORMALIZE P(DET) VALUES DRAW RND U(0,1) SELECT TGT
IS THE OBS THE BLUE SQD LDR
YES
NO
# TGTS > 1
YES
NO
SCHEDULE INDIRECT FIRE
SCHEDULE DIRECT FIRE
CHANGE OBS POSTURE TO PRONE,NONMOVING
RETURN
Figure 7 Select Process

The first task that SELECT does is to check the observer's potential target list and count all detected targets (those with positive probability of detection values). SELECT then normalizes the detected targets' probability of detection values and uses a random Uniform (0,1) variable to select the target to engage. This ensures that targets with the highest probabilities of detection are more likely to be selected. SELECT then schedules a direct fire engagement for a random time (between four and ten seconds in the future). This time accounts for changing observer posture and aiming at the target.

If the observer is the BLUE squad leader and has detected more than two enemy targets, SELECT also schedules an indirect fire event 20 time units in the future. These 20 time units account for the translation of target coordinates into firing data and for preparing and shooting the ammunition. The threshold is set at two enemy targets because fire support is a valuable asset and should only be used for multiple targets and because it is assumed that RED soldiers are moving in formation. Limiting indirect fire requests to the squad leader simulates the fact that only the squad leader has a radio to communicate to the next higher command.

If the observer is a squad leader of either side, SELECT schedules a formation change for 20 time units in the future. This 20 time units allows the squad leader to fire and then decide if his squad needs to change formation and direction to react to the threat posed by his selected target.

Limitations and Assumptions. The target selection process used by SPARTAN has the following limitations and assumptions:

1) If an observer detects a target, he will engage it. There is no criteria for maximum range. This is not unrealistic, as it causes target suppression. In other scenarios, some fire restrictions might be desired.

2) Observers have perfect range estimation for target detection and selection.

Direct Fire Engagements

Unlike the original SPARTAN, the new SPARTAN uses real weapons' data for weapons in the current U.S. and Russian inventories. The default weapons' assignments are listed below:

Table 8 Default Weapons' Assignments

	WEAPON	
POSITION	BLUE	RED
Squad Leader	M16A2	AK-74
Team Leader	M16A2	AK-74
Automatic Rifleman	M249	RPK-74
Grenadier	M203	
Rifleman	M16A2	

Accuracy data was obtained from the Army Material System's Analysis Activity and translated into probability of hit tables for each weapon, range, and target posture. All of the weapons are burst fire weapons, that is, they fire more than one round for each trigger pull. Soldiers do not have the option of firing single shots.

Probability of Hit. The weapons' accuracy data from AMSAA consisted of the horizontal and vertical dispersion factors for aim and ballistic error (in milliradians) for each range from which the weapon was fired. To convert this data into probability of hit tables, I used the Polya-Williams approximation as outlined by Hartman and the Engineering Design Handbook, Army Weapon Systems Analysis, Part I.

Polya-Williams assumes a rectangular target. It also assumes impact distribution is bivariate normal with the mean miss distance in the horizontal and vertical directions and bias equal to zero (13:7-18).

The first step of the approximation was to calculate the total variance in the horizontal and vertical direction. This was done through equations (16) and (17)

$$\sigma^2_x = \sigma^2_{xaim} + \sigma^2_{xballistic} \qquad (16)$$

$$\sigma^2_y = \sigma^2_{yaim} + \sigma^2_{yballistic} \qquad (17)$$

The approximation for a single round hit is then (13:7-18)

$$Phit = \sqrt{x_{term} \times y_{term}} \qquad (18)$$

Where (13:7-18)

$$x_{term} = 1 - e^{\frac{-2 \times L_x \times L_x}{\pi \times \sigma_x \times \sigma_x}} \qquad (19)$$

(Lx is equal to half the target width.)

The equation for the Y term is similar, but with Ly equal to half the target height

$$Y_{term}=1-e^{\frac{-2\times L_y\times L_y}{\pi\times\sigma_y\times\sigma_y}} \qquad (20)$$

Once the single round probability of hit was estimated for the given range and target posture, I assumed independence of each round within the burst. This assumption allowed me to compute the probability of at least one hit for multi-round bursts (10:20-13). Thus, the final probability of hit is

$$Phit_{multi}=1-(1-Phit_{single})^{n} \qquad (21)$$

Where n is equal to the number of rounds fired in the burst (three for each M16 and AK-74 burst and six for each M249 and RPK-74 burst).

Some interpolation was necessary to adjust Phit tables so that the ranges were uniform, but all tables are referenced by range and target posture. The MATHCAD templates for these tables are in Appendix C.

<u>Grenade Launcher</u>. In addition to processing rifle and machine gun engagements, the SHOOT subprogram handles grenade launcher engagements. The first step in the grenade launcher process is to compute the impact point of the round. This is done using accuracy data from AMSAA and two triangular distributions. Two random numbers, representing the horizontal and vertical miss distances from the target

center of mass, are drawn from two triangular distributions with modes of zero and end points equal to plus and minus two standard deviations from the mode. These random numbers are added to the horizontal and vertical coordinates of the target, which gives the impact point of the round.

The probability of hit is determined using Carlton's algorithm (17:78) and is a function of the burst radius of the round (five meters for a M203 round) and the miss distance (target location - impact point)

$$Phit = e^{-\frac{(x_{tgt}-x_{impct})^2+(y_{tgt}-y_{impct})^2}{25}} \quad (22)$$

<u>SPARTAN Direct Fire Process</u>. In this version of SPARTAN, the SHOOT subprogram (Figure 8) processes all direct fire engagements. Although the bullet flight is instantaneous, SHOOT is scheduled between 4 and 10 time units after SELECT to allow time for the observer to assume a firing position and to aim at the target.

The first task of SHOOT is to ensure that the observer is still alive. Next, SHOOT computes the observer-target range and makes a call to the line of sight subprogram to ensure that the observer can still see the target. If LOS is obstructed, the probability of detection value for that target is zeroed out, a search is scheduled (in a random time between 10-20 time units), and program control passes to the next scheduled event.

If LOS still exists, SHOOT then checks to see if the observer has enough ammunition in his current magazine to

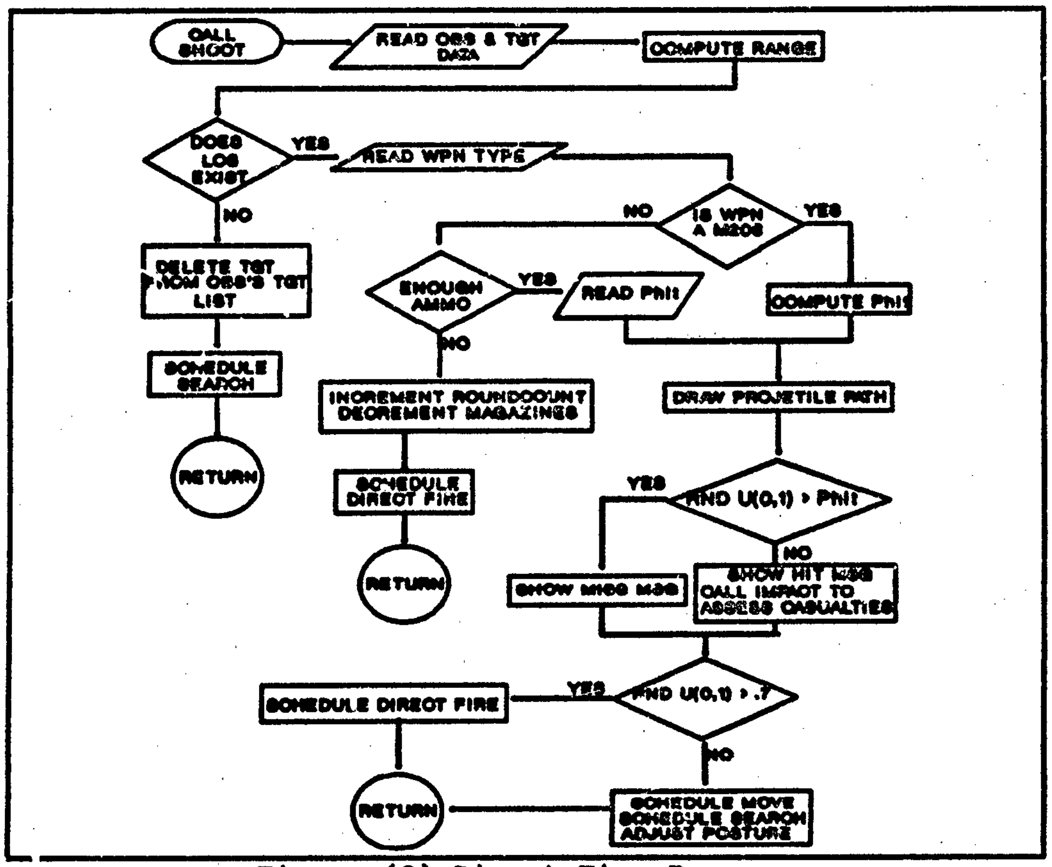


Figure (8) Direct Fire Process

fire a complete burst, if not the observer must change magazines. His rounds remaining count is incremented, magazine count decremented, and a direct fire engagement rescheduled. Program control then passes to the next scheduled event.

If the ammunition count is sufficient, the observer's attribute #6 is updated to reflect the current simulation time, this increases his signature and his probability of being detected by an enemy. Also, the observer's ammunition count is decremented by the number of rounds he fires in a burst (M16A2s and AK74s fire 3 rounds and SAWs and RPK74s fire six rounds per burst). SPARTAN then goes to the correct probability of hit table for the observer's weapon type and draws the Phit value for the target range and posture. Determination of whether the engagement was a hit or a miss is made by a Bernoulli trial.

If the soldier is assigned an M203, SHOOT decides which weapon system he will use to engage the target. If the target is less than 300 meters away, the observer will use his grenade launcher. If the range is too great, he will use his M16. An M203 engagement is represented by a red line drawn from the observer to the impact point. This is accompanied by a single explosion sound representing the round launch. Next, a red circle will be drawn from the impact point to the limits of the burst radius and an explosion will sound. This represents round impact.

Direct fire engagements are represented by a red line drawn from the observer to the target. A burst of sound, representing the number of rounds fired, accompanies the line. If the engagement resulted in a hit, SHOOT makes a call to the impact subprogram and shows a "HIT Phit = ###" message in the upper left corner of the screen. If the engagement was a miss, SHOOT schedules a react to fire for the target in three time units and displays the "MISS Phit = ###" message.

Regardless of the outcome of the engagement, observers have a 30% chance of reengaging a target in five time units and a 70% chance of starting a new move and search cycle.

Determination is made by a random Uniform (0,1) variable. This accounts for shooters not always being able to tell whether or not their engagement was successful. After shooting, observers revert to their prior posture.

Limitations and Assumptions. SPARTAN's direct fire engagement process has the following limitations and assumptions:

1) All soldiers make perfect range estimation.
2) Wind effects on round trajectory are negligible.
3) Linear interpolation of accuracy standard errors is sufficiently accurate for this model.
4) Rounds within a burst of fire are independent.
5) Soldiers will engage targets whenever they acquire them.
6) Bullet flight is instantaneous.
7) Observers do not check to see if squad mates are in the line of fire. The assumption is that they would move or fire around their squad mates.
8) SPARTAN only checks impact versus specified targets. Bullets that miss their intended target disappear. They do not continue on to hit other potential targets.

Indirect Fire Engagements

Unlike the original SPARTAN, the new version models indirect fire systems. The system modeled in SPARTAN is the U.S. Army's 60 millimeter mortar, which is found in light infantry units. SPARTAN models two of these mortars.

Indirect fire is only available to the BLUE side and can only be called by the squad leader. Indirect fire events are scheduled in the SELECT subprogram, whenever the

BLUE squad leader has detected two or more targets. The mortars are located off the screen map at the coordinates X=-500, Y=500 and can range the entire screen map. All missions are immediate suppression missions. This means that each of two mortars fires three rounds. There are no adjust fire missions or repeat missions. Also, all rounds are point detonating, no time delay or aerial burst rounds are modeled.

There are two elements of randomness injected into indirect fire events, target coordinate error and ballistic error. Target coordinate error is caused by the squad leader failing to give the target's actual coordinates to the mortars' Fire Direction Center (the place where firing data is computed). This error has a both a horizontal and a vertical component and can be as much as 100 meters in either direction. The coordinates given to the mortars are computed by equations (23) and (24).

$$x_{coordinate} = x_{tgt} + triag(-100,0,100) \qquad (23)$$

$$y_{coordinate} = y_{tgt} + triag(-100,0,100) \qquad (24)$$

Once the rounds are fired, the ballistic error is computed for each round. Each round has a ballistic error with a vertical and horizontal component, which are randomly drawn from triangular distributions. The modes of both distributions are zero, with the range error being much greater than the deflection error. In reality, mortar fire is a function of tube elevation, charge, air density, round type, and wind. SPARTAN simplifies all these errors to a standard .04 milliradians range standard error and .01 milliradians deflection error. (These figures are actual errors for a 4.2 inch mortar on charge 3 at 100 meters.) To compute these errors I used equations (25) and (26).

$$err_{vertical} = triag(-.08,0,.08) \qquad (25)$$

$$err_{horizontal} = triag(-.02,0,.02) \qquad (26)$$

Because the mortars are located off screen and perpendicular to the screen map's vertical and horizontal planes, each rounds' impact point must be calculated using the trigometric functions (Equations (27) and (28):

$$x_{impct} = x_{tgt} + (err_{ver} \times SIN(\theta) + err_{hor} \times COS(\phi)) \times range \qquad (27)$$

$$y_{impct} = y_{tgt} + (err_{ver} \times COS(\theta) + err_{hor} \times SIN(\phi)) \times range \qquad (28)$$

THETA and PHI are the vertical and horizontal angles of the gun-target line.

Once all rounds' impact points have been calculated, SPARTAN employs the cookie cutter method to calculate the probability of hit for all soldiers (friendly and enemy) within the bursting radius (26 meters) of the rounds.
In SPARTAN's cookie cutter method, the coordinates of a box containing all of the rounds' bursts are determined. Every soldier within this box is declared a casualty. All soldiers within 100 meters of this box are declared suppressed (Figure 9).

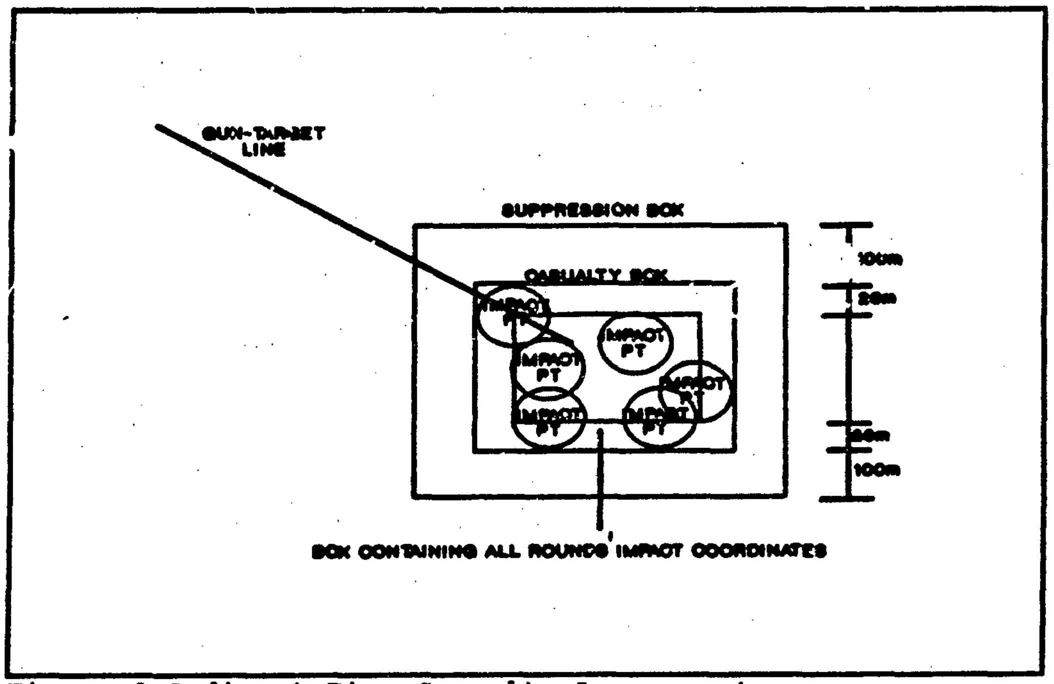
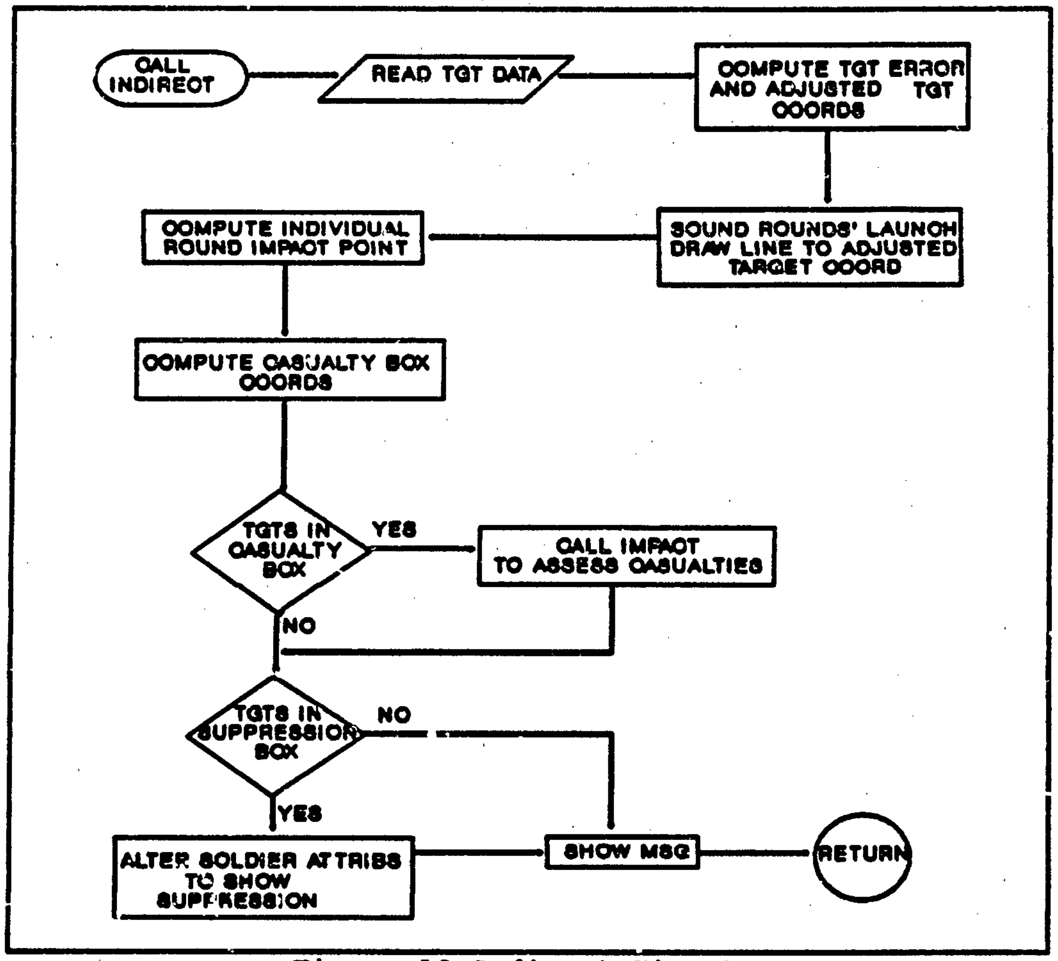


Figure 9 Indirect Fire Casualty Assessment

<u>SPARTAN Indirect Fire Process</u>. The first step of SPARTAN's indirect firing engagement subprogram (Figure 10), INDIRECT, is to determine the grid coordinates of the target that the BLUE squad leader sends to the mortars. SPARTAN uses the grid location of the target selected for direct fire by the squad leader and adds error terms as described above.

SPARTAN then draws a red line to the modified target location from the left edge of the screen. This line represents the gun-target line. SPARTAN then sounds six explosions, these represent the launching of the rounds.

SPARTAN then computes the impact point of each round and plots the detonation on the screen. As impact points are computed, the maximum and minimum vertical and


CALL INDIRECT
READ TGT DATA
COMPUTE TGT ERROR AND ADJUSTED TGT COORDS
SOUND ROUNDS' LAUNCH DRAW LINE TO ADJUSTED TARGET COORD
COMPUTE INDIVIDUAL ROUND IMPACT POINT
COMPUTE CASUALTY BOX COORDS
TGTS IN CASUALTY BOX
YES
CALL IMPACT TO ASSESS CASUALTIES
NO
TGTS IN SUPPRESSION BOX
NO
YES
ALTER SOLDIER ATTRIBS TO SHOW SUPPRESSION
SHOW MSG
RETURN
Figure 10 Indirect Fire Process

horizontal coordinates are determined for use in calculating the casualty and suppression boxes.

The casualty box is then drawn on the screen. The corners of the box are determined by adding 13.5 meters (one-half of the burst radius) to the center of the maximum and minimum impact coordinates. SPARTAN determines if any soldiers are in this box or in the suppression box. An appropriate message is shown on the screen. If casualties are declared, INDIRECT calls IMPACT ASSESSMENT. If soldiers are suppressed, a REACT TO FIRE event is scheduled.

After this, the screen is refreshed to remove the explosions and control of the program passes to the next scheduled event.

Limitations and Assumptions. The limitations and assumptions of the indirect fire engagement process are listed below:

1) Calls for fire are automatic when the BLUE squad leader identifies two or more targets. There might be instances when indirect fire might not be desired.

2) Because mortar accuracy is dependent on so many factors, I simplified the ballistic error. Although the simplification is based on real data (from another mortar type), it is not valid. Mortars are area weapons however, and accuracy is not a overwhelming issue.

3) The cookie-cutter method actually should be employed for each impact, not the sheaf. This simplification was for ease of computation, but still effectively demonstrates the technique.

Impact Assessment

Impact assessment (Figure 11) is not a scheduled event. As SHOOT or INDIRECT subprograms assess a target as being hit, IMPACT is called and the target is immediately assessed as being either killed or wounded. After appropriate attribute updates are made, control then returns back to the SHOOT or INDIRECT subprogram.

Unlike the previous version of SPARTAN, the results of being hit by fire depend on the target's previous status. If the target was unwounded, he has a 30% chance of being killed and 70% chance of being wounded. If the target was already wounded, then his chances of being killed increase

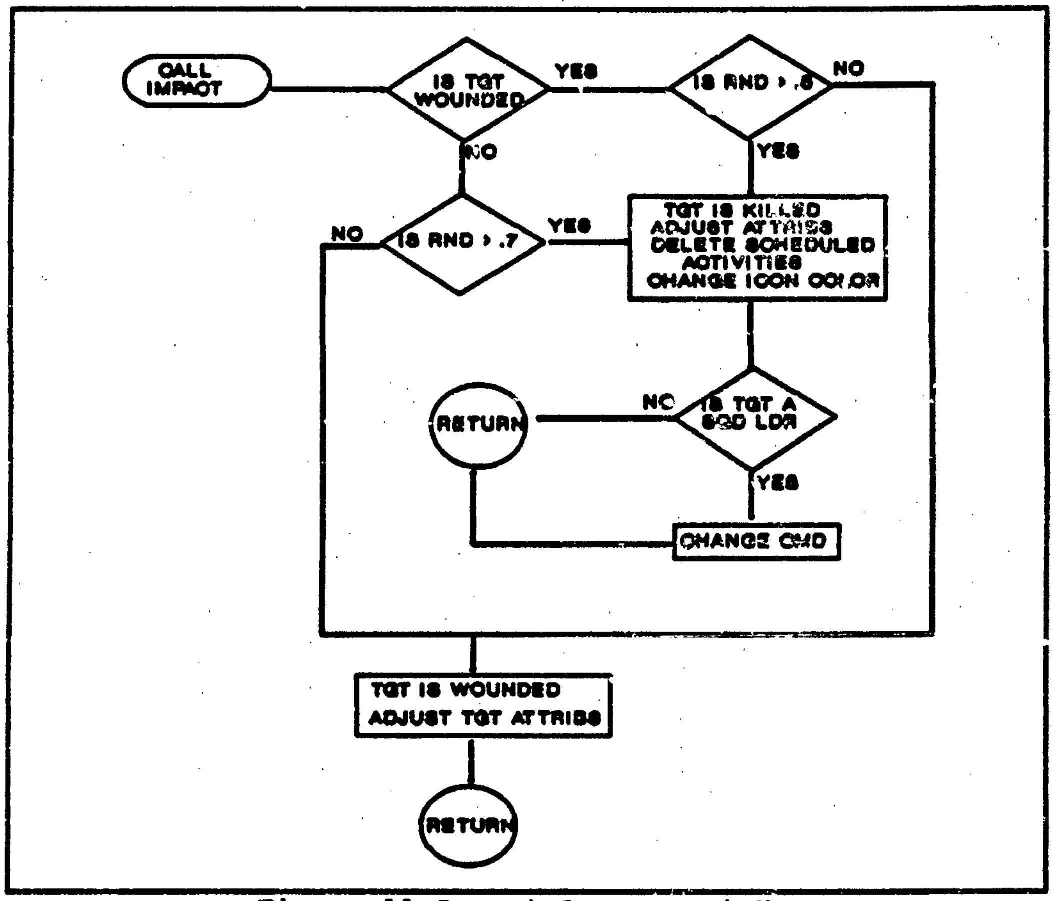

CALL IMPACT
IS TGT WOUNDED
YES
NO
IS RND > .5
NO
YES
IS RND > .7
NO
YES
TGT IS KILLED ADJUST ATTRIBS DELETE SCHEDULED ACTIVITIES CHANGE ICON COLOR
IS TGT A SQD LDR
NO
YES
RETURN
CHANGE CMD
TGT IS WOUNDED ADJUST TGT ATTRIBS
RETURN

Figure 11 Impact Assessment Process

to 50%. The determination of casualty type is made by a random draw from the Uniform (0,1) distribution.

If the target is declared to be wounded, his wound status is updated and he is put in a prone nonmoving status. If the target is killed, his wound status is updated, he is put in a prone nonmoving status, his icon is changed to a grey color, and all of his scheduled events are pulled from the event calendar.

If a killed target is a squad leader, then a change of command is affected. SPARTAN searches for a designated team leader and updates his position attribute to reflect that he is now a squad leader.

There are a few assumptions and limitations associated with this process:

1) Once a target has been wounded, his ability to absorb punishment does not change. His chance of being killed remains at 50%.

2) Although dead soldiers are prone and nonmoving, they can still be engaged. Observers do not discern between alive, wounded, and dead targets.

3) Impact assessment is obviously a function of weapon type and impact point. The probabilities of outcomes referenced above are not based on any valid data.

React to Fire

Reaction to fire is a scheduled event in SPARTAN. It is scheduled for three time units after a direct fire or a indirect fire event results in a miss. Soldiers react to fire in a stochastic manner, depending on their current status.

Table 9 Reaction to Fire Probabilities

CURRENT POSTURE	REACTION	PROBABILITY
Standing	Crouch	Pr = .5
	Prone	Pr = .2
	Prone/nonmoving	Pr = .1
	No Change	Pr = .2
Crouching	Prone	Pr = .4
	Prone/nonmoving	Pr = .1
	No Change	Pr = .5
Prone	Prone/nonmoving	Pr = .5
	No Change	Pr = .5

NOTE: Unless the reaction specifies a move status, the target's move status is unchanged.

Reactions are decided by a random Uniform (0,1) variable. The values assigned to the reaction probabilities are not based on valid data, but seem plausible for the given tactical situation.

If the targeted soldier is a squad leader, then his entire squad emulates his reactive posture and move status. This is in line with the "follow me and do as I do" command the squad leader gives to his squad.

Change Formation and Direction

The last type of scheduled event in SPARTAN is the formation and direction change. This event is scheduled during the SELECT target subprogram, whenever a squad leader selects a target. The thought process behind this event is that the squad leader might want to evaluate the worth of changing the formation and direction of movement of his squad to bring more firepower to bear on the target that he perceives as being most dangerous.

Change Formation/Direction Process. The first function of the DIRECTION subprogram (Figure 12) is to assess whether a formation and direction change is warranted. SPARTAN does this by determining the observer-target angle from the squad leader to his target. If this observer-target line is less than 25 degrees off the current azimuth of travel, only a formation change is directed. In this case, all soldiers' position attributes are updated to reflect the new formation. If the difference between these angles is

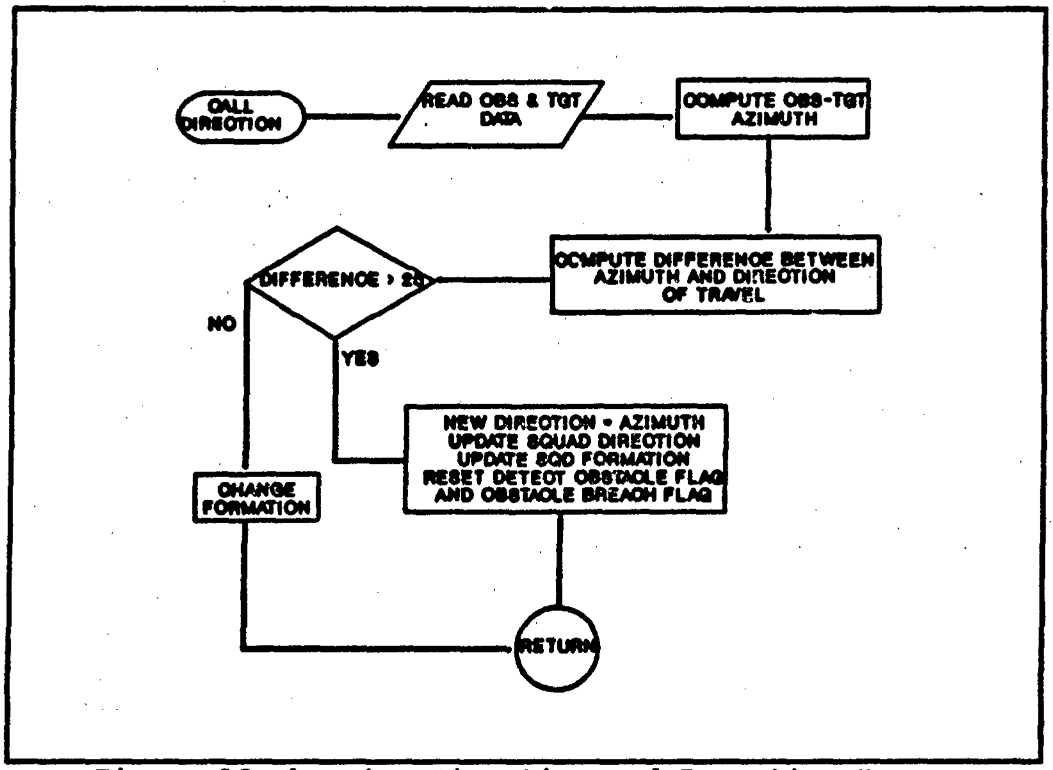


Figure 12 Changing Direction and Formation Process

greater than 25 degrees, then the direction of all the soldiers on the squad leader's side are changed to the azimuth of the observer-target line. A formation change is also directed. A message reflecting the side and formation change is shown the upper left screen corner. The new formation is "squad on line, teams in wedges". This formation allows all soldiers to bring fire to bear on the squad leader's target.

After a formation and direction change, a flag is tripped and the squad leader must once again search for wire obstacles in his new direction. Also, the obstacle breached flag is tripped and any obstacles that the squad encounters, must be breached.

<u>Limitations and Assumptions</u>. There are several limitations and assumptions associated with the logic of this process:

1) Location changes are instantaneous. Graphics are not updated to reflect location changes. During the next MOVE event, soldiers might move more than 20 meters. This is not unrealistic as soldiers must run to adjust formations in real life.

2) Squads sometimes change formations during breaching operations. Because they are in a nonmove status, a REFRESH is scheduled.

3) As ranges close, squads change direction more frequently as squad leaders select targets that are on opposite sides of the enemy formation.

Command and Control

SPARTAN has no specific subprogram designed for modeling command and control. Instead, the entire simulation was designed to reflect the realities of small unit combat and the control that the squad leader asserts over his squad. Thus, command and control logic and limitations were imbedded in all SPARTAN processes.

The squad leader dictates movement speed and direction for his squad. This models the formation movement that small units maintain and the fact that the squad leader controls the formation.

The squad leader directs formation changes based on his perceptions. He also is the one with the radio and thus has the capability for calling for fire. The call for fire is contingent on the squad leader's acquiring multiple targets.

The squad leader also controls how his squad moves. If the squad leader reacts to fire by hitting the prone position and ceasing movement, then his entire squad does also.

Summary

In this chapter, I have outlined the methods that SPARTAN uses to model various combat processes. The intent was to provide a sufficient level of detail for the reader to gain a understanding of the equations involved in the processes, the algorithms and decisions made in the processes, how the processes interact, why a particular technique was chosen for implementation, and some of the limitations and assumptions of the processes.

Chapter V. Conclusion

Introduction

This chapter concludes the SPARTAN thesis. The first section summarizes the purpose and results of the thesis work. This is followed by a list of suggested improvements to SPARTAN.

Summary

This thesis effort was aimed at improving SPARTAN, a high resolution land combat model demonstrator, and making it a more useful tool for land combat modeling courses. Although the list is not exhaustive, the major improvements of this version of SPARTAN over the original are listed below:

1) Greater terrain detail. Forested areas are incorporated in the new SPARTAN. They affect movement and target acquisition.

2) Obstacles. The old SPARTAN did not have obstacles. The current version allows the user to input one linear obstacle.

3) Movement. The movement of entities is now tied to formations. The rate of movement and placement of soldiers is determined by the squad leader. Also, the squad leader can alter the formation and direction of travel based on his target selection.

4) Target acquisition. The acquisition process more closely resembles the NVEOL process used in the Army's current generation of high resolution combat models.

5) Weapon's data. Weapon's accuracy data is from the Army Material System's Analysis Activity (AMSAA) and is the same data used by JANUS and CASTFOREM.

6) Weapon types. SPARTAN now has a default weapons' mix of five different weapons, all currently used by the U.S. or Russia.

7) Indirect fire. SPARTAN now also models indirect fire, both mortars and grenade launchers.

8) Command and control. Command and control is modeled in greater detail than in the previous version.

9) Reaction to fire. Soldiers' reaction to fire is modeled in greater detail and more realistically than in the previous version.

10) Preprocessor. The preprocessor is much more user friendly. It not only allows the user to view data and alter it for new scenarios, but also serves as an instructional tool for terrain modeling and for terrain familiarization.

11) Graphics. Graphics are greatly improved over the previous version. The screen battlefield resembles a map, which improves comprehension of model activities. Also the icons and entity activities are more clearly delineated, improving understanding of model activities.

In making these improvements, the objectives of portability, usability, simplicity, and applicability guided all modeling decisions.

SPARTAN is portable. Written in QuickBASIC and then compiled, its data files and two executable programs total less then 500 K. Thus, it can be carried on all high density disks and can be used on most IBM/IBM clone computers.

SPARTAN is usable. Users of SPARTAN need only minimal knowledge of computers and modeling techniques and no knowledge of infantry tactics. The preprocessor and simulation are designed to teach them about the former subjects. The menu format also facilitates use by less knowledgeable individuals.

SPARTAN is simple. QuickBASIC reads almost like English, thus SPARTAN's computer code is very easy to understand. Also, the structured programming makes following model processes easy. The structured programming also makes the possibility of future improvements more likely.

SPARTAN is also applicable. All processes were modeled on techniques used in the current generation of high resolution land combat models.

The result of meeting the design objectives and the improvements listed above was a model which demonstrates the following:

1) Time keeping and an implementation technique for event set management and synchronization.

2) Algorithms used to model movement, target acquisition and detection, target selection, weapon selection, weapon accuracy, direct and indirect fire attrition, line of sight, reaction to fire, and simple command and control decisions and processes.

3) Stochastic techniques for representing the occurrence of random events and outcomes on the battlefield.

4) Data requirements for model components.

5) An example of the components for a typical combat model such as scenario input, a preprocessor, the simulation model, various types of output, and accompanying documentation.

Recommendations

Although SPARTAN is a finished model and is capable of being employed for its intended use, there are several improvements that could be made to further enhance its value as a learning tool. Because of the structured programming design of SPARTAN and programming language (QuickBASIC), making these improvements should be an easy task. Some of these suggested improvements are:

1) Design a replication loop. Although designed as an analytical model, SPARTAN currently only executes a simulation once. Building an outer replication loop, with the capacity to store data from each simulation run, could improve its value for simulation study.

2) Improve the movement process. Most high resolution models use movement control points to direct movement. SPARTAN entities only change direction in engagements. Implementing a movement control point type process would enhance scenarios.

3) Incorporate some sort of user interaction. Allow users to direct formation changes or direction changes.

4) Design more terrain data bases. This involves drawing the map and filling in the terrain data array. The user could pick a terrain option in the preprocessor.

5) Allow users to alter acquisition tables in the preprocessor. Allow the user to alter the sky-ground brightness ratios and target background contrasts to simulate limited visibility conditions.

6) Increase the number of entities allowable in order to simulate entire squads on both sides.

7) Incorporate indirect fire for both sides.

8) Incorporate vehicles in the simulation (and also anti-vehicle weaponry).

9) If user interaction is incorporated, use hidden icons. Do not display RED icons until they are acquired by BLUE soldiers.

This is only a partial list of recommended improvements. There are many possible ideas, but the original scope of the modeling project was limited to producing a teaching tool, not a new video game.

Conclusion

This modeling effort provides the military modeling community with a high resolution land combat model demonstrator. Following the model development process as outlined by Pritsker, a model was developed that is simple, usable, portable, and applicable. By using SPARTAN with its algorithms based on those used in the current generation of high resolution models and its extensive help files and other documentation, beginning modelers should gain a great deal of insight about the uses and limitations of combat models.

Appendix A: Threshold Pinf Template

This appendix contains the template used for creating the threshold probability of detection given infinite time to observe the target area. The technique of assigning such observer-target threshold values is used by both JANUS and CASTFOREM. Using this technique, each possible observer-target pairing is assigned a random threshold level of Pinf. This is then compared to the deterministically derived actual Pinf to see if tha target is giving off significant signature to be detected by that observer. This template was written using MATHCAD 2.5 software. It shows the calculations used to create the random cor level and then displays a partial array of threshold Pinf values.

This MATHCAD template computes the random threshold Pinf value for every observer target pair. It uses the triangle distribution function to simulate the normal population called for by JANUS documentation (17:25-27).

JANUS assigns a random threshold Pinf value to each observer-target pair, using a lognormal distribution whose underlying normal distribution has a mean of 3.5 and a standard deviation of .698 (17:28). To replicate this normal distribution, I use a triangle function with a mode of 3.5, a lower bound equal to two standard diviations below the mode (2.1) and an upper bound equal to 4.896.

i and j are the indices $i := 1 .. 12$ $j := 1 .. 12$

a is the lower bound $a := 2.1$

b is the upper bound $b := 4.896$

d is the mode $d := 3.5$

Since SPARTAN has 12 soldiers, there are 132 possible observer-target combinations. For ease of matrix manipulation, I will round out the matrix to 144 (12 rows and 12 columns).

r is the 12 x 12 matrix of random Uniform (0,1) variables

$$r_{i,j} := \mathrm{rnd}(1)$$

Now define some relations of the upper and lower bounds and the mode.

e = mode - lower bound $e := d - a$

f = upper bound - lower bound $f := b - a$

g = upper bound - mode $g := b - d$

Next assign random numbers from the triangle distribution function (based on the previously assigned random U(0,1) variable) to the x matrix (12 x 12):

$$x_{i,j} := \mathrm{if}\left[r_{i,j} \le \frac{e}{f}, a + \left[e \cdot f \cdot r_{i,j}\right]^{.5}, b - \left[g \cdot f \cdot \left[1 - r_{i,j}\right]\right]^{.5}\right]$$

Since the random threshold is a lognormal variable, substitute the exponential function of the random triangle distribution.

$$X_{12 \cdot (i-1)+j} := e^{x_{i,j}}$$

This matrix becomes the matrix of threshold cor values for condition I of target acquisition.

To test the distribution, I plotted a histogram of the threshold values.

$X2 := \mathrm{sort}(X)$

k is the number of hisotgram cells

$$k := (1 + 3.3 \cdot \log(144)) \qquad k = 8.123$$

Since k > 8, I round it up to 9 $\quad k := 9$

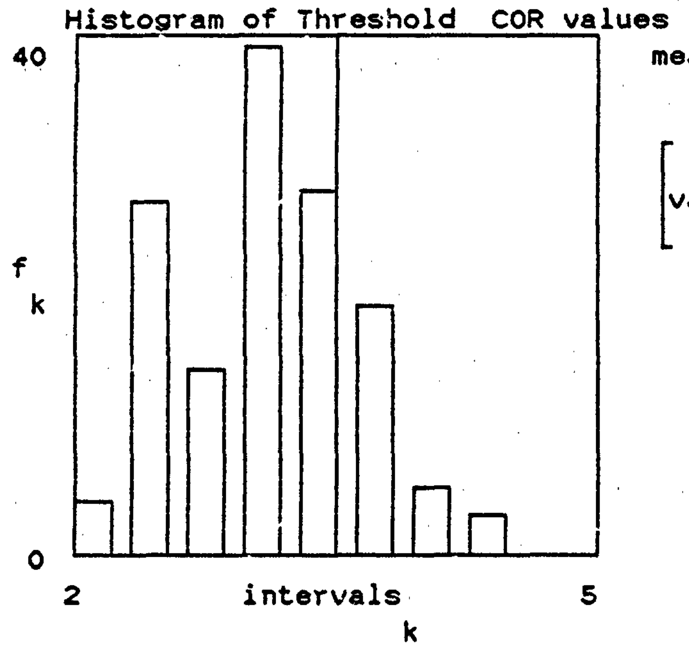
c defines the histogram cell width

$$c := \frac{\max(X) - \min(X)}{k} \qquad c = 0.325 \qquad \frac{f}{8} = 0.35$$

$k := 1 \,..9$

$\mathrm{intervals}_k := 2.1 + c \cdot (k - 1)$

$f := \mathrm{hist}(\mathrm{intervals}, X2) \quad f_9 := 0$



$\text{mean}(X) = 3.292$

$$\left[\text{var}(X)\cdot\frac{144}{143}\right]^{.5} = 0.585$$

This plot looks like a lognormal plot.

Remember that the mean number of resolvable cycles for detection at any range is 3.5. The mean of 3.292 is pretty close to this. Also the Standard Error of .585 is close to the standard error .698

Now transfer the random cor values into a 12 x 12 matrix

$$x_{i,j} := X_{12\cdot(i-1)+j}$$

To save time in the simulation, transfer the cor values into Pinf values using the equation below:

$$pinf1_{i,j} := \frac{\left[\frac{x_{i,j}}{3.5}\right]^{2.7+.7\cdot\frac{x_{i,j}}{3.5}}}{1+\left[\frac{x_{i,j}}{3.5}\right]^{2.7+.7\cdot\frac{x_{i,j}}{3.5}}}$$

This Pinf table becomes the threshhold level that observers must exceed in order to meet condition I of the acquisition process.

Below is listed one column of threshold Pinf values:

$pinf1_{i,1}$
0.164
0.18
0.627
0.431
0.434
0.543
0.42
0.261
0.555
0.253
0.301
0.51

Appendix B: Probability of Acquisition Template

This appendix contains the template used to create the cycles resolvable by the observer tables and the probability of detection given infinite observation time (Pinf) tables. All calculations are based on techniques used in the NVEOL model as employed by JANUS and CASTFOREM. The appendix goes through the calculations and assumptions and then shows the cycles resolvable table and the Pinf table for targets not in wooded areas.

The following MATHCAD 2.5 template computes the acquisition tables using the NVEOL model algorithms. There are three conditions for target acquisition:

1) The target is giving off sufficient signature for the observer to detect him.
2) The observer has line of sight to the target.
3) The observer is looking at the target during the specified search cycle time.

This template computes tables for use in calculating meeting Condition I i.e. cycles resolvable by the observer and the Pinf (probability of detection given unlimited time to look at the target). To save computation time in the simulation, these values are computed in increments of 100 meters.

I. Attenuated Target Contrast

The first step of the NVEOL target acquisition model is to identify the range and the attenuated target contrast. The latter is a function of target-background contrast, sky-ground background contrast, range, and an atmospheric i is defined as the array location pointer $i := 1 ..10$

r is the array of ranges from 100m to 1000 m (in km) $r_i := \frac{i}{10}$

Now calculate the attenuated target contrast using the equation (17:26):

$$\text{attenuatedcontrast} := \frac{\text{targetcontrast}}{1 + 2.5\cdot(\exp(r\cdot.01) - 1)} \;\square$$

where 2.5 is the sky-ground background contrast for a bright day. The JANUS documentation did not provide any values for the atmospheric attenuation coefficient, so after experimentation, I used the value .01.
Target contrast values range between .2 and .3. I used .3 for targets in wooded areas and .29 for targets not in wooded areas.

ac1 is the target attenuated contrast for targets in wooded terrain.

$$ac1_i := \frac{.3}{1 + 2.5 \cdot \left[\exp\left[r_i \cdot .01\right] - 1\right]}$$

ac2 is the target attentuation contrast for targets not in wooded terrain

$$ac2_i := \frac{.29}{1 + 2.5 \cdot \left[\exp\left[r_i \cdot .01\right] - 1\right]}$$

These equations result in the attenuated target contrasts below:

Target in wooded area

$$ac1 = \begin{bmatrix} 0.299 \\ 0.299 \\ 0.298 \\ 0.297 \\ 0.296 \\ 0.296 \\ 0.295 \\ 0.294 \\ 0.293 \\ 0.293 \end{bmatrix}$$

Target not in wooded area

$$ac2 = \begin{bmatrix} 0.289 \\ 0.289 \\ 0.288 \\ 0.287 \\ 0.286 \\ 0.286 \\ 0.285 \\ 0.284 \\ 0.284 \\ 0.283 \end{bmatrix}$$

II. Cycles Resolvable by the Observer

The cycles resolvable by the observer is a sixth degree polynomial function of the attenuated contrast, the range (in km) and the minimum target dimension. First compute the sixth degree polynomial:

$k := 0 \,..5$

$$cor1_i := \sum_k \left[\ln\left[ac1_i\right]^{6-k}\right]$$

This gives the cycles per milliradian for targets not in wooded areas.

$$cor2_i := \sum_k \left[\ln \left[ac2_i \right] \right]^{6-k}$$ This gives the cycles per milliradian for targets in wooded areas.

To get the actual cycles resolvable, the numbers computed above must be multiplied by the minimum target dimension and divided by the range. This gives cycles per target dimension by range.

target posture	height	width	minimum dimension
standing	1.8m	.8m	.8m
crouching	.9m	.8m	.8m
prone	.45	.8m	.45m

Remembering the heuristics of doubling the dimension if the target has fired during the last 20 time units or is moving the target posture array becomes:

$$tgtpos := (.8 \quad .45 \quad 1.6 \quad .9)$$

Now finish computing the tables of cycles resolvable by the observer:
$j := 1 \,..4$

$$cor1_i := \frac{1}{r_i} \cdot cor1_i \qquad cor2_i := \frac{1}{r_i} \cdot cor2_i$$

$cor1a := cor1 \cdot tgtpos$
$cor2a := cor2 \cdot tgtpos$

Now we have the tables for cycles resolvable by the observer for various ranges and target postures.

Cycles resolvable on a target in a wooded area:

$$\begin{array}{l|cccc|l} \text{POSTURE} & \text{Standing Crouch} & \text{Prone} & \text{Standing Crouch} & \text{Prone} & \text{Range} \\ & 9.116 & 5.128 & 18.231 & 10.255 & 100m \\ & 4.646 & 2.614 & 9.293 & 5.227 & 200m \\ & 3.157 & 1.776 & 6.314 & 3.552 & 300m \\ & 2.413 & 1.357 & 4.826 & 2.715 & 400m \\ & 1.967 & 1.106 & 3.934 & 2.213 & 500m \\ \text{cor1a} = & 1.67 & 0.939 & 3.339 & 1.878 & 600m \\ & 1.458 & 0.82 & 2.915 & 1.64 & 700m \\ & 1.299 & 0.731 & 2.598 & 1.461 & 800m \\ & 1.176 & 0.661 & 2.351 & 1.323 & 900m \\ & 1.077 & 0.606 & 2.154 & 1.212 & 1000m \end{array}$$

The last two columns designate target who have just fired or are moving.

Cycles resolvable on a target not in a wooded area:

$$\begin{array}{l|cccc|l} \text{Posture} & \text{Standing Crouch} & \text{Prone} & \text{Standing Crouch} & \text{Prone} & \text{Range} \\ & 11.701 & 6.582 & 23.402 & 13.163 & 100m \\ & 5.954 & 3.349 & 11.907 & 6.698 & 200m \\ & 4.038 & 2.272 & 8.077 & 4.543 & 300m \\ & 3.081 & 1.733 & 6.163 & 3.467 & 400m \\ & 2.508 & 1.411 & 5.015 & 2.821 & 500m \\ \text{cor2a} = & 2.125 & 1.196 & 4.251 & 2.391 & 600m \\ & 1.853 & 1.042 & 3.705 & 2.084 & 700m \\ & 1.648 & 0.927 & 3.297 & 1.854 & 800m \\ & 1.49 & 0.838 & 2.979 & 1.676 & 900m \\ & 1.363 & 0.767 & 2.726 & 1.533 & 1000m \end{array}$$

These numbers look reasonable, despite the educated guesses on several coefficients. (The average cor for target identification is 3.5.)

III. Computing Pinf

The last step is to calculated Pinf, the probability of target detection given unlimited time to search for the target. This is only shown for targets not in wooded areas:

$$\mathrm{pinf2}_{i,j} := \frac{\left[\dfrac{\mathrm{cor2a}_{i,j}}{3.5}\right]^{2.7+.7\cdot\frac{\mathrm{cor2a}_{i,j}}{3.5}}}{1+\left[\dfrac{\mathrm{cor2a}_{i,j}}{3.5}\right]^{2.7+.7\cdot\frac{\mathrm{cor2a}_{i,j}}{3.5}}}$$

This gives us the results below:

	Standing		Standing		
Posture	Crouch	Prone	Crouch	Prone	Range
	0.998	0.927	1	0.999	100m
	0.888	0.463	0.998	0.932	200m
	0.623	0.204	0.974	0.719	300m
	0.396	0.105	0.902	0.492	400m
	0.256	0.062	0.791	0.331	500m
pinf2 =	0.174	0.041	0.666	0.23	600m
	0.124	0.029	0.549	0.166	700m
	0.093	0.021	0.45	0.124	800m
	0.072	0.016	0.37	0.097	900m
	0.057	0.013	0.308	0.077	1000m

This data was fed into the Pinf tables INFW.dat and INFNW.dat (for INFinte in the Woods and INFinte Not in the Woods).

Appendix C: Probability of Hit Template

This appendix contains the template used to create probability of hit tables for all direct fire weapons modeled in SPARTAN. Weapons' data is based on actual firing data obtained from the Army Material System's Analysis Activity (AMSAA). The technique uses the Polya-Williams approximation and the negative binomial function to derive probabilities of hit for various ranges. This appendix shows the raw accuracy data, demonstrates the calculations, and shows the final hit probabilities for all weapons systems.

The following MATHCAD 2.5 template computes the probability of hit for a multi-round burst of fire using M16A2 accuracy data.

I. Background

To compute the multi-round probability of hit for a burst fire weapon, I used the Polya-Williams Approximation suggested by Hartman in his unpublished lecture notes on High Resolution Modeling.

The Polya-Williams Approximation assumes the target is rectangular and is perpendicular to the gun-target line. It also assumes the impact distribution is bivariate normal with the means of the dispersions (both vertical and horizontal) and the bias equal to zero. Using it the Pr(hit) is (13:7-18):

$$\mathrm{Phit} := \sqrt{X \cdot Y} \quad \square$$

where

$$X := 1 - e^{-2 \cdot \frac{Lx^2}{\pi \cdot \sigma_x^2}} \quad \square \qquad (2{*}Lx = \text{width of target})$$

$$Y := 1 - e^{-2 \cdot \frac{Ly^2}{\pi \cdot \sigma_y^2}} \quad \square \qquad (2{*}Ly = \text{height of target})$$

and σx and σy are the standard errors (dispersions) in the Horizontal and Vertical directions.

II. Data

Data for all weapons systems is from the U. S. Army Material

Systems Analysis Activity (AMSAA). All data is from man-in-the-loop tests. Both firers and targets were stationary. Firers shot from the prone unsupported position, then data about dispersions of impacts from center mass aim points was collected. The M16A2 data is presented below:

Weapon Type:	M16A2
Round:	M855 Ball
Ranges:	25, 50, 100, 200, 300, 400, 500, 600 meters

PRONE UNSUPPORTED POSITION

DISTANCE	BALLISTIC ERROR		AIM ERROR	
	HORIZONTAL σ	VERTICAL σ	HORIZONTAL σ	VERTICAL σ
25 m	7.85 mils	8.98 mils	12.00 mils	12.00 mils
50 m	7.85 mils	8.98 mils	7.2 mils	7.2 mils
100 m	7.85 mils	8.98 mils	4.7 mils	4.7 mils
200 m	7.85 mils	8.98 mils	3.5 mils	3.5 mils
300 m	7.85 mils	8.98 mils	3.1 mils	3.1 mils
400 m	7.85 mils	8.98 mils	2.9 mils	2.9 mils
500 m	7.85 mils	8.98 mils	2.8 mils	2.8 mils
600 m	7.85 mils	8.98 mils	2.7 mils	2.7 mils

To make accessing the probability of hit tables easier, I extrapolated the accuracy data so that hit probabilities are in increments of 100 meters from 100 to 800 meters.

This results in the dispersion table below:

SPARTAN M16A2 accuracy data

DISTANCE	BALLISTIC ERROR		AIM ERROR	
	HORIZONTAL σ	VERTICAL σ	HORIZONTAL σ	VERTICAL σ
100 m	7.85 mils	8.98 mils	4.7 mils	4.7 mils
200 m	7.85 mils	8.98 mils	3.5 mils	3.5 mils
300 m	7.85 mils	8.98 mils	3.1 mils	3.1 mils
400 m	7.85 mils	8.98 mils	2.9 mils	2.9 mils
500 m	7.85 mils	8.98 mils	2.8 mils	2.8 mils
600 m	7.85 mils	8.98 mils	2.7 mils	2.7 mils
700 m	7.85 mils	8.98 mils	2.6 mils	2.6 mils
800 m	7.85 mils	8.98 mils	2.5 mils	2.5 mils

III. Calculations

The first step in computing the Phit is to define the arrays of input data:

$$range := \begin{bmatrix}100\\200\\300\\400\\500\\600\\700\\800\end{bmatrix} \quad \sigma balx := \begin{bmatrix}7.85\\7.85\\7.85\\7.85\\7.85\\7.85\\7.85\\7.85\end{bmatrix} \quad \sigma baly := \begin{bmatrix}8.98\\8.98\\8.98\\8.98\\8.98\\8.98\\8.98\\8.98\end{bmatrix} \quad \sigma aimx := \begin{bmatrix}4.7\\3.5\\3.1\\2.9\\2.8\\2.7\\2.6\\2.5\end{bmatrix}$$

$\sigma aimy := \sigma aimx$

Next, compute total error in the horizontal and vertical directions using the equation below:

$$\sigma^2 := \sigma_{aim}^2 + \sigma_{bal}^2 \ \square$$

At the same time, convert the variance in milliradians to meters.

One milliradian deviation is equal to one meter of error from center target at 1000 meters. Therefor to convert mils to meters, multiply mils by km.

We now compute the total variance
$i := 1 \,..8$

$$\sigma 2x_i := \left[\sigma balx_i \cdot \frac{range_i}{1000}\right]^2 + \left[\sigma aimx_i \cdot \frac{range_i}{1000}\right]^2$$ = total variance in the horizontal direction for each range.

$$\sigma 2y_i := \left[\sigma baly_i \cdot \frac{range_i}{1000}\right]^2 + \left[\sigma aimy_i \cdot \frac{range_i}{1000}\right]^2$$ = total variance in the vertical direction for each range.

The standard error in meters at different ranges is

Horizontal Error $\sqrt{\sigma 2x_i}$	Range	Vertical Error $\sqrt{\sigma 2y_i}$
0.915	100m	1.014
1.719	200m	1.928
2.532	300m	2.85
3.347	400m	3.775
4.167	500m	4.703
4.981	600m	5.626
5.789	700m	6.544
6.591	800m	7.457

Assume that targets (personnel) are rectangular and approximately .8m X 1.8m. There are three target postures: standing, crouching/kneeling, and prone. The width of the target will not change based on target posture, but the height will. The height factors are:

$$j := 1..3 \qquad posture := \begin{bmatrix} 1 \\ .5 \\ .25 \end{bmatrix} \begin{matrix} \text{Standing} \\ \text{Crouching} \\ \text{Prone} \end{matrix}$$

Now compute our X values. Since

$$width := 2 \cdot Lx \; \square \quad \text{where width} = .8$$

Define $Lx := .4$

Using the Polya-Williams approximation:

$$X_i := 1 - e^{-2 \cdot \frac{Lx^2}{\pi \cdot \sigma 2x_i}}$$

$$X = \begin{bmatrix} 0.115 \\ 0.034 \\ 0.016 \\ 0.009 \\ 0.006 \\ 0.004 \\ 0.003 \\ 0.002 \end{bmatrix}$$

Now compute the Y values. There will be three Y values at each distance because of target posture. Given the target height (y) is 1.8m, we get the following values for Ly at various postures and ranges:

$$Ly_{i,j} := \frac{1.8}{2} \cdot posture_j$$

$$Ly = \begin{array}{c} \begin{array}{ccc} \text{stand} & \text{crouch} & \text{prone} \end{array} \\ \begin{bmatrix} 0.9 & 0.45 & 0.225 \\ 0.9 & 0.45 & 0.225 \\ 0.9 & 0.45 & 0.225 \\ 0.9 & 0.45 & 0.225 \\ 0.9 & 0.45 & 0.225 \\ 0.9 & 0.45 & 0.225 \\ 0.9 & 0.45 & 0.225 \\ 0.9 & 0.45 & 0.225 \end{bmatrix} \end{array}$$

Using the Polya-Williams approximation:

$$Y_{i,j} := 1 - e^{-2 \cdot \frac{Ly_{i,j}^{2}}{\pi \cdot \sigma 2y_{i}}}$$

Now the Y values are computed, compute the single round Phit

$$Phit_{i,j} := \sqrt{X_{i} \cdot Y_{i,j}}$$

Target Posture

Phit =

stand	crouch	prone	Range
0.213	0.116	0.059	100m
0.066	0.034	0.017	200m
0.031	0.016	0.008	300m
0.018	0.009	0.005	400m
0.012	0.006	0.003	500m
0.008	0.004	0.002	600m
0.006	0.003	0.002	700m
0.005	0.002	0.001	800m

Now, assume the rounds of the burst are independent and compute the probability of at least one hit given multiple rounds fired in the burst (3 rounds for the M16A2)(10:20-14)

$$p3hit_{i,j} := 1 - \left[1 - Phit_{i,j}\right]^{3}$$

This results in the table below, which is used by SPARTAN to determine hit probability for the M16A2 (firing a three round burst) for various ranges and target postures.

Target Posture

p3hit =

stand	crouch	prone	
0.512	0.31	0.168	100m
0.186	0.099	0.05	200m
0.091	0.047	0.024	300m
0.053	0.027	0.014	400m
0.034	0.017	0.009	500m
0.024	0.012	0.006	600m
0.018	0.009	0.005	700m
0.014	0.007	0.003	800m

IV. AK-74

Using the same method as for the M16A2, I computed the multi-round probability of hit for the Russian AK-74

Weapon Type:	AK-74
Round:	5.45mm
Ranges:	25, 100, 200, 300, 400, 500, 600, 700, 800 meters

PRONE UNSUPPORTED POSITION

DISTANCE	BALLISTIC ERROR		AIM ERROR	
	HORIZONTAL σ	VERTICAL σ	HORIZONTAL σ	VERTICAL σ
25 m	8.47 mils	8.9 mils	12.0 mils	12.0 mils
100 m	8.47 mils	8.9 mils	4.7 mils	4.7 mils
200 m	8.47 mils	8.9 mils	3.5 mils	3.5 mils
300 m	8.47 mils	8.9 mils	3.1 mils	3.1 mils
400 m	8.47 mils	8.9 mils	2.9 mils	2.9 mils
500 m	8.47 mils	8.9 mils	2.8 mils	2.8 mils
600 m	8.47 mils	8.9 mils	2.7 mils	2.7 mils
700 m	8.47 mils	8.9 mils	2.6 mils	2.6 mils
800 m	8.47 mils	8.9 mils	2.6 mils	2.6 mils

Dropping the 25m data, this data resulted in the table of hit probabilities below (for a three round burst):

TARGET POSTURE	STANDING	CROUCHING	PRONE
RANGE			
100m	.493	.297	.161
200m	.176	.093	.048
300m	.086	.044	.022
400m	.05	.025	.013
500m	.032	.016	.008
600m	.023	.011	.006
700m	.017	.009	.004
800m	.013	.007	.003

IV. M249 and RPK-74 (Squad Automatic Weapons)

Using the same method as for the M16A2, I computed the multi-round probability of hit for both US and Russian SAWS (they both had the same accuracy data).

Weapon Type:	M249 (US) and RPK-74 (Russian)
Round:	5.56mm and 5.45mm
Ranges:	50, 300, 600, 900, 1200, 1500, 1800, 2100 meters

BIPOD SUPPORTED POSITION

DISTANCE	BALLISTIC ERROR		AIM ERROR	
	HORIZONTAL σ	VERTICAL σ	HORIZONTAL σ	VERTICAL σ
50 m	1.27 mils	1.41 mils	7.1 mils	7.1 mils
300 m	1.27 mils	1.41 mils	3.1 mils	3.1 mils
600 m	1.27 mils	1.41 mils	2.7 mils	2.7 mils
900 m	1.27 mils	1.41 mils	2.6 mils	2.6 mils
1200 m	1.27 mils	1.41 mils	2.5 mils	2.5 mils
1500 m	1.27 mils	1.41 mils	2.5 mils	2.5 mils
1800 m	1.27 mils	1.41 mils	2.4 mils	2.4 mils
2100 m	1.27 mils	1.41 mils	2.4 mils	2.4 mils

Because SPARTAN only models a maximum range of 1000 meters, I interpolated for shorter ranges and dropped longer ranges, giving the altered accuracy data below:

DISTANCE	BALLISTIC ERROR		AIM ERROR	
	HORIZONTAL σ	VERTICAL σ	HORIZONTAL σ	VERTICAL σ
100 m	1.27 mils	1.41 mils	6.1 mils	6.1 mils
200 m	1.27 mils	1.41 mils	4.1 mils	4.1 mils
300 m	1.27 mils	1.41 mils	3.1 mils	3.1 mils
600 m	1.27 mils	1.41 mils	2.7 mils	2.7 mils
900 m	1.27 mils	1.41 mils	2.5 mils	2.5 mils
1000 m	1.27 mils	1.41 mils	2.5 mils	2.5 mils
1800 m	1.27 mils	1.41 mils	2.4 mils	2.4 mils
2100 m	1.27 mils	1.41 mils	2.4 mils	2.4 mils

This results in the probability of hit table below (six round burst):

TARGET POSTURE	STANDING	CROUCHING	PRONE
RANGE			
100m	.958	.828	.581
200m	.826	.603	.368
300m	.725	.488	.283
600m	.34	.19	.1
900m	.189	.1	.051
1000m	.156	.082	.042
1800m	.055	.028	.014
2100m	.04	.02	.01

Appendix D: SPARTAN Operating Instructions

General

This appendix is a user's manual for SPARTAN and can be used (in conjunction with the SPARTAN and STARTUP help screens) to run the SPARTAN simulation. This manual and the HELP screens are designed for use by students who have at least a limited knowledge of IBM PCs and high resolution land combat modeling.

SPARTAN is a two-sided high resolution land combat model originally developed by Army CPT Dave Cox (AFIT GOR92-M) as an instructional tool for use in a land combat modeling course. It is intended to demonstrate current modeling techniques as used by the Army's present generation of high resolution land combat models. All algorithms and most of the data are representative of those used by the Army's two premier models, JANUS and CASTFOREM. Tactical formations, weapons' mixes and accuracy data, and decision rules are as accurate and realistic as the author could make them based on ten years of infantry experience and current Field Manuals. SPARTAN, however, makes no claim as to being a true replication of reality. In keeping with the intended purpose of the model, decision rules are simplistic and time representation is adjusted so the model looks right. No effort has been make to perform any validation of SPARTAN as an analytic tool.

This User's Manual is organized into four sections. Section I is a discussion of SPARTAN construction and modeling processes. Section II includes a discussion of the default scenario and the existing default data files. Section III contains operating instructions for STARTUP (the preprocessor). Section IV provides operating instructions for the model. More information about these topics can be obtained through the HELP screens in the preprocessor and SPARTAN and in the SPARTAN thesis.

Section I. Model Description

SPARTAN is designed primarily like an analytical model of a force on force conflict. The combat is between two opposing squad sized elements in a 1000m x 1000m area. The conflict takes place in rolling terrain that is primarily open, but that has some forested areas. The combat is during daylight and obscuration is not a factor.

SPARTAN represents the following combat processes: movement, target search, target selection, weapon selection and direct fire engagement, indirect fire engagement, reaction to fire, impact assessment, command and control, and obstacle breaching. All processes are resolved at the individual soldier level. No processes are aggregated.

SPARTAN allows human participation in only two places, the preprocessor, where the user can alter data files to fit his scenario, and in the terminating conditions, which the user can alter before beginning the simulation run. Once the simulation begins, no user action impacts on model outcomes.

SPARTAN uses an event scheduling technique to synchronize activities and maintain time representation within the model. Future events are maintained on the event list which lists the events by type, actor, and time (time being a generic time unit not related to minutes or seconds). Only nine types of events are put on the calendar, these are listed in the table below:

Table 10 SPARTAN's Scheduled Processes

NUMBER	EVENT
1	Target Search
2	Target Select
3	Direct Fire Engagement
4	Move
5	End Breaching Operation
6	React to Fire
7	Change Direction and Formation
8	Refresh the Screen
9	Indirect Fire

All other events are instantaneous (the simulation clock does not advance when they are occurring). SPARTAN does not use linked lists to keep track of calendar events. As a scheduled event is completed, SPARTAN checks the calendar for the next event with the lowest start time and executes that event (after showing a message as to current simulation time, event type and event actor in the upper right corner). SPARTAN continues to schedule and execute events until the event list is empty or one of the terminating conditions is met.

One modification to this event step process allows users the option of letting the simulation execute as fast as the computer can process events or delaying event processing by setting the ratio of simulation time to clock time (to a maximum of 1 to 5). This option can be turned off by altering the terminating conditions.

Force Composition. SPARTAN models a maximum of 12 soldiers divided into two sides, RED and BLUE. Soldiers on the screen are represented by stick figures of the appropriate color. Squad leaders of either side are denoted by a box drawn around the figure. Soldiers' capabilities and status are captured by the 15 attributes in the soldier file (see Section III for a complete list of soldier attributes). The default scenario has 9 BLUE soldiers organized and equipped as U.S. infantry and three RED soldiers organized and equipped in the old Soviet style. Organization and weapons are listed in Table 11.

Table 11 Default Force Composition

POSITION	WEAPON	
	BLUE	RED
Squad Leader	M16A2	AK-74
Team Leader	2 x M16A2	AK-74
Automatic Rifleman	2 x M249	RPK-74
Grenadier	2 x M203	
Rifleman	2 x M16A2	

In addition to the squad's organic weapons, the BLUE side has the advantage of the use of two 60mm mortars. These mortars are not represented on the screen and are notionally located off the screen map.

<u>Terrain Representation</u>. The terrain used in SPARTAN is a one kilometer square area loosely based on a U.S.Army 1:50,000 scaled map of Germany. Terrain is represented by a 50 x 50 system of square grid cells. Each cell has a horizontal (east-west) coordinate and a vertical (north-south) coordinate numbered from 1-50. Each cell also has an elevation, a mobility factor, and a visibility factor. These attributes allow the model to represent terrain relief features and to adjust movement and target detection.

The SPARTAN screen is read just like any military map. The lower left corner is GRID 000 000 (read X coordinates first and Y coordinates second) and the upper right corner is GRID 1000 1000. Magenta lines are spaced every 200m to aid in distance referencing during the simulation. (These lines might not appear to be square because of how different monitors and computer models break down horizontal and vertical resolution.) The colors on the map are those used on military maps (Table 12).

TABLE 12 MAP COLOR TRANSLATION

COLOR	REPRESENTS
Blue	water
Green	forests, wooded areas
Red	manmade objects (roads)
Black	roads
Brown	Contour intervals

The contour interval for this map is ten meters. Elevations range from 60 to 110 meters. To provide more realistic representation of terrain, mobility, visibility and elevation are not constant in like areas. For example, visibility and mobility decrease as you go deeper into wooded areas and elevation rises as you get closer to next higher contour interval. Roads and streams have no impact on movement and are only provided for user reference.

Movement. When a soldier's attributes are set for movement, he will move in a direction and speed designated by his squad leader. All moves are in 20m increments, with the movement time for these increments varying with soldier posture and the mobility factor of the terrain cell in which the movement starts. The squad leader designates the movement speed for his entire squad.

Search. SPARTAN uses a continuous search process based on the Night Vision Electro-Optical Laboratory (NVEOL) model used by JANUS and CASTFOREM. In SPARTAN this is a continuous process, with every soldier conducting a 360 degree search every 20-40 time units. Soldiers search only for enemy soldiers, therefore there is no need for identifying friends or foes.

In order for target detection to occur three conditions must be met:

1) The target must give off sufficient signature to be detected by the observer.

2) The observer must have line of sight to the target.

3) The observer must be looking at the target.

The NVEOL model is based on something called number of resolvable cycles. A resolvable cycle is a pair of contrasting light and dark panels laid across the minimum target dimension. Different numbers of resolvable cycles are required for different levels of target detection. For example, identifying that something is there requires less cycles than identifying that something is a specific type of tank.

SPARTAN, like JANUS and CASTFOREM, assigns every possible target-observer pair a random threshold probability of detection given time (Pinf). This has the effect of making some observers' acquisition of some targets easier than others and injects probability into the acquisition process. To detect a target, the observer must be able exceed the threshold level of probability of detection for that target. Both of these numbers are in lookup tables. The threshold level is referenced for each target-observer pair and the deterministically derived Pinf is referenced by target posture, range, and background contrast.

If the target is giving off sufficient signature or if the observer has already tried and failed to detect the target, SPARTAN checks the line of sight from the observer to the target. Line of sight (LOS) can be thought of as a line drawn from the observer's eye to the eye of the target. If no intervening terrain or vegetation breaks this line, then SPARTAN assumes the entire target can be seen; there is no partial line of sight. SPARTAN checks LOS in every terrain cell between observer and target.

If LOS exists, SPARTAN determines if the observer was looking at the target during that particular search event. A random time is drawn and the probability of detection is calculated. A Bernoulli trial determines if the target was detected. If the target was acquired, the target is added to the observer's potential target list, a SELECT TARGET event is scheduled, and a fraction of the probability of detection is added to the other squad members' potential target list to simulate intra-squad communication about the target. Also, a message is shown in the upper left corner, telling that either a RED or a BLUE soldier has detected enemy at some grid coordinate. If the target was not detected, a fraction of the probability of detection is added to the observer's potential target list to simulate already searching that area once. This increases the probability of detection in subsequent searches.

<u>Target Selection</u>. Once one or more targets has been detected by a soldier during a search cycle, a TARGET SELECT is scheduled. All the probabilities of detection for the observer's potential targets are normalized and a random Uniform(0,1) variable decides which target will be engaged. That target is added to the observer's attribute list (attribute #14) and a DIRECT FIRE engagement is scheduled. If the observer is the BLUE squad leader and he has two or

more potential targets, an INDIRECT FIRE event is also scheduled.

Direct Fire Engagement. Direct fire engagements begin with a check to ensure that LOS still exists between target and observer. If LOS still exists, SPARTAN ensures that the observer has sufficient ammunition to engage the target. IF not, SPARTAN changes magazines (decrements magazine count and increments the round count) and reschedules a direct fire engagement. If LOS exists and the observer has ammunition, then the observer shoots at the target.

SPARTAN represents direct fire by a red line drawn from the observer that closes on the target. A burst of sound that represents the number of rounds fired accompanies the line. In SPARTAN, M16s and AK74s fire three round bursts, while Squad Automatic Weapons (SAWs) fire six round bursts.

The M203 gunner makes a decision as to whether fire his M203 or his M16. If the range is less than 300 meters, the gunner uses his grenade launcher. This is represented by a red line connecting the observer and the impact point, a red circle drawn at the point of impact, and an explosion sound. The screen is then refreshed.

The results of direct fire engagements are determined by a Bernoulli trial. For all weapons except the M203, a random number is drawn and compared to probability of hit tables that are referenced by weapon, range, and target posture. The hit/miss results of the M203 engagement are determined using the Carlton method where the probability of hit is determined by the ratio of the miss distance to the burst radius of the round. If at least one round hits the target, an IMPACT ASSESSMENT is scheduled. After an engagement, a hit/miss message and the probability of hit will appear in the upper left corner.

After an engagement, the observer has a 30% chance of reengaging the target or a 70% chance of moving and searching again.

Indirect fire Engagements. Only the BLUE side has indirect fire capabilities in SPARTAN. The system replicated is the light infantry company's 60mm mortar. In SPARTAN, the two mortars are located off screen at the position X = -500 Y = 500. Mortars are called by the BLUE squad leader whenever he identifies two or more enemy soldiers. There is an associated time delay between calls for fire and when the mortars actually fire to account for computation of firing data and the preparation of ammunition. All missions are immediate suppression, both tubes firing three rounds (six rounds total). There are no repeat or adjust fire missions. All rounds are high explosive point detonating.

There are several errors built into indirect fire missions. First, the grid coordinates that the squad leader calls to the tubes can be off as much as 100 meters in either the vertical or horizontal direction, this replicates map reading error. There is also the ballistic error of the rounds. In both cases, a triangle distribution is used to simulate normal distributions.

Indirect fire engagements are represented by an irregular burst of six explosions representing the rounds' launch. A red line is then drawn from the tubes to the referenced target coordinate. Six red circles and accompanying explosions then denote rounds impacting. Next, a black rectangle is drawn over the red circles. This rectangle represents the area in which all personnel, friendly or enemy, are assessed as being hit. Outside this rectangle (to a distance of 200m), all personnel are assessed as suppressed. This method of casualty assessment is known as the cookie cutter method.

Impact Assessment. If a target is assessed as being hit as result of either a DIRECT or INDIRECT FIRE engagement, an IMPACT ASSESSMENT is called. If the target previously was uninjured, it has a 30% chance of being killed and a 70% chance of being wounded. If the target was already wounded, his chances of being killed increase to 50%. If the target is wounded, his attributes are updated to reflect that he is now prone and in a nonmoving status. If the target is killed, his icon changes to grey, his wound status, position status and move status attributes are updated, and all his future events are removed from the event list. If the target was a squad leader, a succession of command to one of the designated team leaders takes place.

Reaction to Fire. Targets react to fire based on their current posture and move status (Table 13). If the reacting soldier is a squad leader, all squad members adopt his new posture.

Table 13 React to Fire

CURRENT STATUS	REACTION	
Standing$x	Crouching	(Pr=.5)
	Prone	(Pr=.2)
	Prone/nonmoving	(Pr=.1)
	No change	(Pr=.2)
Crouching	Prone	(Pr=.4)
	Prone/nonmoving	(Pr=.1)
	No change	(Pr=.5)
Prone	Prone	(Pr=.5)
	Prone/nonmoving	(Pr=.5)

NOTE: Unless move status is specifically indicated in the reaction, it does not change from the current status.

Obstacles. The default data files contain no obstacles, but users can input one obstacle in the preprocessor. If an obstacle is emplaced, than a flag is tripped and the squad leader checks along his current azimuth to see if he can identify the obstacle (at which point a message reflecting obstacle identification will appear in the upper left corner). If the obstacle does not intersect the squad leader's azimuth, then SPARTAN assumes the entire squad can bypass it.

No direction change is caused by obstacle identification, instead the squad will continue to move forward until the first squad member hits the obstacle.

When the squad hits the obstacle, breaching commences. Breaching takes 100 time units, during which the squad assumes a prone nonmoving status (although the squad might change formation if it comes under fire while breaching). At the end of 100 time units, the squad posture changes back to standing and moving.

Formation and Direction Changes. If the squad leader selects a target that is more than 25 degrees off the current azimuth, he will direct a formation change to bring his squad on line and a direction change towards the new target in order to focus more fire power. If the target is less than 25 degrees off the current azimuth, only a formation change is directed. In either case, squad members' position attributes are updated, but the current scheduled moves are not altered. This will be most noticeable when it appears that observers are shooting at empty spaces or that empty spaces are engaging targets. For this reason, a REFRESH screen is scheduled during breaching operations. Otherwise, the graphic's discrepancies will self correct during the next MOVE cycle.

Command and Control. SPARTAN has no specific command and control module, instead command and control is built into almost every aspect of the simulation. The squad leader dictates movement speed (this maintains formation integrity). The squad leader also dictates the posture of all squad members (although in the absence of guidance they will react individually). The BLUE squad leader also is the only one who can call for indirect fire. Both squad leaders adjust formations and direction of movement to attack the enemy they feel is most dangerous. The squad leader is also the only soldier who can identify obstacles. Finally, if the squad leader is killed, there is a succession of command based on subordinate leaders.

Output and Help. SPARTAN provides extensive Help and Output. Help menus are accessible through the main menu and provide more specific information on algorithms and equations used in modeling combat processes. Output is available both during program execution and at program completion. See Section IV for more specifics.

Section II. Default Scenario

T$xs section provides information about the default SPARTAN scenario.

Situation. The SPARTAN land combat model replicates two squad sized forces fighting in terrain that is not controlled by either side. Both forces have roughly equivalent types of small arms, although BLUE is a bigger force. BLUE also has indirect fire. This situation is like many low intensity conflict scenarios.

Terrain. The terrain in the example is mostly rolling grassy farmland. Elevation varies between 60 and 110 meters. There is one stream that is fordable to dismounted soldiers and several wooded areas which slow movement and hinder observation. There are two all weather capable roads and numerous farm trails in the area. Wire obstacles from previous operations are in the area.

<u>Weather</u>. Weather is not expected to hinder operations for either side. A clear sunny day is expected. Neither side has obscurants.

<u>Mission</u>. BLUE conducts a combat patrol to identify and destroy any RED forces in the area of their patrol. RED forces seek to deny BLUE forces access to the area. BLUE has superior firepower, but RED is willing to accept proportionally higher casualties. BLUE is successful if they destroy two thirds or more of RED, forcing RED to withdraw. RED is successful if they kill one third or more of BLUE, forcing them to withdraw. Neither side has been in the area before and there are no prepared positions.

<u>Equipment</u>. Equipment for the default scenario is the same as identified in Section I.

<u>Data files</u>. Ten data files are required for this scenario. For a complete list, see Section IV.

<u>Section III. STARTUP</u>

This section contains information about STARTUP.exe, a preprocessor that can be used to load, view, or edit default data files. Basically, STARTUP.exe reads the .dat extension data files into arrays, allows the user to edit them, and creates .exp files for SPARTAN to read.

STARTUP is menu driven and like SPARTAN has extensive help files. The four data files that STARTUP allows the user to edit are the terrain data file, the soldier attribute list, the initial event list, and the probability of hit tables.

Terrain Editor. Unlike the first version of SPARTAN, the terrain attribute list is hard wired. Because the data must match the graphical representation on the screen for the screen to be meaningful, the data files are closed to users. The terrain editor, however, is designed to assist first time land combat modelers. It offers the options below:

1) View map--allows the user to view the map.

2) Add obstacles--allows the user to add one wire obstacle. Creates OBS file for SPARTAN.

3) View terrain dat--allows the user to view terrain cell data.

4) View elevation data--allows the user to see how terrain cell elevation data dictates contour lines.

5) Line of Sight--allows the user to pick observer location and checks the line of sight for user input ranges and fields of view.

The terrain data file is call MAP1.dat and contains 3 attributes for each of 2500 terrain cells. Each cell has an elevation attribute between 60 and 110, a mobility factor between .1 and 1 (1 being unimpeded mobility), and a visibility factor between .5 and 1 (1 is unimpeded visibility).

Soldier Attribute Editor. The Soldier Attribute Editor allows the user to accomplish tasks listed below:

1) View BLUE soldier attributes--allows the user to view selected BLUE soldier attributes.

2) View RED soldier attributes.

3) Add soldiers--allows the user to add soldiers (for a maximum of 12 soldiers).

4) Delete soldiers.

5) Edit soldier attributes--allows the user to edit a selected soldier's attributes.

6) Pick formation and location--allows to pick the BLUE squad leader's location and one of four BLUE formations. Automatically updates position data for the remainder of the squad.

Each soldier has 15 attributes. These are listed in Table 14.

Table 14 Soldier Attributes

ATTRIBUTE	DESCRIPTION	RANGE OF VALUES
1	side	1=BLUE -1=RED
2	duty position	1=SL, 2=ASL, 3=GRNDR, 4=AR, 5=Rifleman
3	horizontal coord	0 - 1000
4	vertical coord	0 - 1000
5	# grenades	0 - 32
6	time last fired	
7	posture before direct fire engagement	
8	direction of travel	0 - 6.28 radians
9	move status	1 = moving 0 = stationary
10	posture	1=standing, 2=crouching 3=prone
11	weapon	1=M16A2, 2=AK74, 3=SAW 4=M203, 5,6=user defined
12	rounds/magazine	M16=30, AK74=40, SAW=200
13	# magazines	M16=6, AK74=6, SAW=3
14	target selected	0 - 12
15	wound status	0=dead, 1=wounded, 2=alive

It is not necessary to start all soldiers on the game map, but it is recommended that at least the squad leader start on the screen. SPARTAN will carry soldiers and do their computations as if soldiers were on the board, but will not draw them until they are completely on the screen.

Probability of Hit Editor. The Probability of Hit Editor allows the user to either review current Phit tables or to create his own. SPARTAN uses the Polya-Williams approximation to compute the single round hit probability of a hit on a rectangular target. A negative binomial function is then used to compute Phit for bursts of fire.

Polya-Williams needs both vertical and horizontal aim and ballistic error. STARTUP will show the raw error data and the computed Phit for each weapon referenced by target posture and range.

Users can also input their own weapons data, but they need aim and ballistic error(vertical and horizontal) for ranges of 100 to 800 meters (in 100 meter increments). These errors must be measured in radians. STARTUP will then compute and show the Phit tables. For the table to be used however, the user must alter at least one soldier's weapon (attribute #11) to reflect the new weapon type. The first user weapon is designated as weapon type five and the second as weapon type six.

Event List Editor. The Event List Editor allows the user to accomplish tasks listed below:

1) View initial event list.

2) Add events to the initial event list.

3) Delete events from the initial event list.

The event list has an event type, event time, and event actor for all scheduled events. The default event list has a move and a search event for all soldiers (for a total of 24 events). The user must be careful not to delete these as these events initiate all other actions. In addition, the squad leader for either side must be the first soldier for that side to execute a move (in order to establish a move time). Otherwise, the move time is zero and the first soldier to move will continue to march until he goes off the screen. Users can add events (for a total of 45 events) using the editor.

Section IV. Set Up and Use of SPARTAN

This section provides instructions for running SPARTAN.

Hardware. SPARTAN actually consists of two separate executable files and ten default data files. STARTUP.exe is a preprocessor designed to allow users to preview data files and to edit them as they desire to alter the scenario. SPARTAN.exe is the executable simulation. Both programs were written in QuickBASIC and were compiled to create executable files that can run on any IBM (or clone) with DOS 2.1 or better and at least EGA capable monitor. The program also needs a minimum of 512K memory. Altering SPARTAN code requires use of QuickBASIC.

Data Files. Data files are in ASCII format. Most were originally made using MathCad 2.5 and transferred to the QUICKBasic directory for use by SPARTAN. There are ten data files required to run the preprocessor, these are listed in Table 15.

Table 15 Default Data Files

FILENAME	DESCRIPTION
map1.dat	terrain data file (50x50x3)
event.dat	initial event list (99x3)
soldat.dat	soldier attribute list (12x15)
M16.dat	M16 P(hit) tables (8x3)
AK74.dat	AK74 P(bit) tables (8x3)
SAW.dat	SAW P(hit) tables (8x3)
cor.dat	P(acquisition) tables (10x3)
INFNW.dat	P(detect) tables (10x3)
INFNW.dat	P(detect not in woods) tables (10x3)
THRESHOLD.dat	target-observer detect levels (12x12)

These data files are accessible only by STARTUP.exe. This preprocessor reads these files, allows the user to alter some of them, and then creates files with .exp extension. The .exp extension files are the ones read by SPARTAN. SPARTAN requires 13 data files (Table 16).

Unless the user creates data during the preprocessor run, three of these files are empty: p5hit.exp, p6hit.exp and obs.exp. These are the files that the user has for creating his own weapon P(hit) files (for a maximum of two) and an obstacle file.

Operating SPARTAN. To operate SPARTAN follow the instructions listed below:

STEP 1. Ensure all ten default data files (TABLE 13), three empty files, SPARTAN.exe and STARTUP.exe are in the current directory.

Table 16 SPARTAN Data Files

FILENAME	DESCRIPTION
map1.exp	terrain data file (50x50x3)
event.exp	initial event list (99x3)
joe.exp	soldier attribute list (12x15)
p1hit.exp	M16 P(hit) tables (8x3)
p2hit.exp	AK74 P(hit) tables (8x3)
p3hit.exp	SAW P(hit) tables (8x3)
cor.dat	P(acquisition) tables (10x3)
INFNW.dat	P(detect) tables (10x3)
INFNW.dat	P(detect not in woods) tables (10x3)
THRESH.dat	target-observer detect levels (12x12)
p5hit.exp	user input weapon's accuracy phit tables
p6hit.exp	user input weapon's accuracy phit table
obs.exp	obstacle data

STEP 2. At the command prompt, type "startup". STARTUP.exe presentation screen will appear with a brief message of explanation about STARTUP. You can access STARTUP's help menu from either the main menu or by hitting <F1> at any time during program execution. In STARTUP.exe, the user can either load the default files or view and edit them. First time users should use the load default files option first to ensure all necessary data files for SPARTAN are created before editing any files. Once the user is finished in STARTUP, exit the program.

STEP 3. Ensure files listed in Table 16 are in the directory with the executable files. For first time users, running the Load Default Files option in the preprocessor will ensure that all files are present. At the command prompt, type "spartan". The SPARTAN.exe presentation screen will appear. When the user hits <CR>, a brief explanation about SPARTAN will appear. The simulation will begin loading data files when the user hits <CR> again.

STEP 4. SPARTAN will then query the user about altering terminating conditions. Default terminating conditions are passage of 350 time units, 3 BLUE KIA, 2 RED KIA, or after 5000 calendar events. The user can also change the random number seed and turn off the slaving of simulation time to passage of real time.

STEP 5. After the user alters terminating conditions (or elects not to), the simulation begins. The user can bring up the display menu at any time by hitting <F1> or can refresh the screen by hitting <F2>. The display menu allows the user to:

1) View current soldier attributes.
2) View current potential target list.
3) View events currently on calendar.
4) View current Battle Statistics.
5) Refresh the screen.
6) Call the Help Menu.
7) Resume the simulation.
8) Terminate the simulation.

STEP 6. When the simulation terminates, SPARTAN queries the user about viewing output. If the user elects to do so, SPARTAN will show the final soldier attribute list, potential target list, event calendar, Battle Statistics, and Kill Card. SPARTAN also makes a history.dat file which contains a chronological listing of calendar events, actors, and times. If this file is not renamed, SPARTAN will overwrite it the next simulation run.

Summary

This User's Guide has provided information about SPARTAN processes, the default scenario, and how to run SPARTAN and its preprocessor. It is not intended as a stand alone document. Much more information about algorithms is provided in the programs' help files and in the thesis. Since this is a first draft, comments from users about this guide and the model will be gratefully accepted (and selectively implemented) to improve SPARTAN's worth as a teaching tool.

Appendix E: Preprocessor Code

This appendix provides a listing of the QuickBASIC 4.5 program code for the preprocessor, STARTUP.exe. STARTUP is a menu driven program that allows the user to view, edit, and alter the default data files containing the soldier attributes, initial event list, and obstacle list. STARTUP also allows the user to create his own weapon type, provided the user has accuracy data for that weapon. Specific information about these files is contained in Chapter III and in the user's manual (Appendix D).

STARTUP is a single module, with subprograms for each program function. All subprograms are listed in their entirety, except for the help files. These files are screens of formatted information that appears elsewhere in this thesis.

QuickBASIC does not have a line continuation feature, so ampersands (&) have been used to indicate a line extension. Comment lines are indicated by a single quotation mark (').

```
'**************************************************************
'*                                                            *
'*                         STARTUP.bas                        *
'**************************************************************

'PURPOSE:  This program is a preprocessor for the SPARTAN
'combat model.  It allows the user to view, modify, or
'create his data files.

'This section of the program defines the subprograms and the
'variables passed to the subprograms when the program is
'called.

DECLARE SUB aboutspartan ()'formatted information about
                           'SPARTAN history, development,
                           'and data files
DECLARE SUB explain ()     'Explains how to use the menu
DECLARE SUB phit ()        'Menu for Phit editor
DECLARE SUB cphit ()       'Prints current raw accuracy data
                           'and phit tables
DECLARE SUB addwpn ()      'accepts user input accuracy data
DECLARE SUB hitdefault ()  'loads default phit data
DECLARE SUB elist ()       'Event Editor Menu
DECLARE SUB help ()        'Help Menu
DECLARE SUB joehelp ()     'Soldier Attribute Editor Help
DECLARE SUB maphelp ()     'Terrain Editor Help
DECLARE SUB phithelp ()    'Phit Editor help
DECLARE SUB evnthelp ()    'Event Editor help
DECLARE SUB mapp ()        'Terrain File Editor menu
DECLARE SUB SOLDIER ()     'Soldier Attribute Editor
DECLARE SUB joeatrib ()    'edits soldier attributes
DECLARE SUB red (opt%, r)  'displays red soldiers' attributes
DECLARE SUB blue (opt%, b) 'displays blue soldiers' attributes
DECLARE SUB add ()         'adds soldiers to soldier list
DECLARE SUB delete ()      'deletes soldiers
DECLARE SUB format ()     'allows user to pick BLUE formation
                           'and location
DECLARE SUB mapdefault ()  'loads default terrain data
DECLARE SUB TERRAINDAT ()  'displays terrain data file
DECLARE SUB contour ()     'displays cell elevation within
                           'contour intervals
DECLARE SUB wire ()        'allows user to input a wire
                           'obstacle
DECLARE SUB map (opt%)     'Draws map
DECLARE SUB los ()         'shows line of sight for
                           'user input data
DECLARE SUB opening ()     'opening screen
DECLARE SUB default ()     'loads default data files
                           'Frame draws the frame for
                           'different presentation screens
DECLARE SUB frame (left%, right%, top%, bottom%, fore%, back%)
DECLARE SUB editevnt (opt, num)  'edits events
DECLARE SUB addevnt ()     'adds events
DECLARE SUB delevnt ()      'Allows users to delete events

'Dynamic allows the computer to create data arrays outside the
'64K allocated for executable programs.
'$DYNAMIC

'This section dimensions data arrays
DIM SHARED mapl(50, 50, 3) 'contains terrain cell data
DIM SHARED soldat(12, 15)  'contains soldier attribute values
DIM SHARED event(99, 3)     'contains initial event list
DIM SHARED lin(10, 4)       'contains obstacle data
DIM SHARED p1(8, 4)         'contains M16A2 Phit data
DIM SHARED p2(8, 4)         'contains AK74 Phit data
DIM SHARED p3(8, 4)         'contains SAW Phit data
DIM SHARED p4(8, 4)         'contains user input Phit data
DIM SHARED p5(8, 4)         'contains user input Phit data

CLS                         'clears the screen
COLOR 15, 9                 'establishes blue as the screen
                            'background color and white as the
                            ' foreground color

CALL opening  'calls opening screen
CALL explain  'calls expainatory screen

DO      'queries the user for the next task until he is done
COLOR 15, 9
CLS

'This next line establishes options for frame on the main menu
left% = 10: right% = 70: top% = 3: bottom% = 24: fore% = 15:
&back% = 9
CALL frame(left%, right%, top%, bottom%, fore%, back%)
LOCATE 4, 30: PRINT "MASTER MENU"
LOCATE 6, 25: PRINT "1) Work on terrain file"
LOCATE 8, 25: PRINT "2) Work on soldier file"
LOCATE 10, 25: PRINT "3) Work on weapon P(hit) file"
LOCATE 12, 25: PRINT "4) Work on event file"
LOCATE 14, 25: PRINT "5) Read Help file"
LOCATE 16, 25: PRINT "6) Input default data"
LOCATE 18, 25: PRINT "7) Exit program"
LOCATE 20, 25: PRINT "Input your selection number"
9 LOCATE 21, 25: ch$ = INPUT$(1)

SELECT CASE ch$
CASE "1"
  CALL mapp
CASE "2"
  CALL SOLDIER
CASE "3"
  CALL phit
CASE "4"
  CALL elist
```

```
CASE "5"
  CALL help
CASE "6"
  CALL default
CASE "7"
  EXIT DO
  CASE ELSE                        'this is the error trap
   BEEP
   LOCATE 22, 25: PRINT "Try again PYLE, choices are between
&   1 and 7"
   GOTO 9
END SELECT
LOOP
CLS
END


SUB aboutspartan
'**************************************************************
'ABOUTSPARTAN is an information screen that provides the
'history of SPARTAN, its development, and its data
'requirements. The formatted text is not presented here, but
'can be found in the user's guide.
'**************************************************************

CLS
CALL frame(10, 70, 4, 8, 15, 9)
LOCATE 6, 26: PRINT " ABOUT SPARTAN"
END SUB


SUB add
'**************************************************************
'The subprogram add allows the user to add soldiers to the
'soldier list. The program first checks to ensure that the
'number of soldiers you desire to input does not exceed the
'maximum allowable of 12. The program then calls to either
'the BLUE or RED subprograms, which will display the current
'soldier list and then allow the user to input the new
'soldier's attributes.
'**************************************************************
'VARIABLES: r = number of red soldiers to be added
'           b = number of blue soldiers to be added
'           empty = max number of allowable additions
'           total = number of total desired additions

CLS
CALL frame(10, 70, 4, 15, 15, 9)
LOCATE 5, 20: INPUT "How many RED soldiers do you wish to
&  add?"; r
LOCATE 6, 20: INPUT "How many BLUE soldiers do you wish to
&  add?"; b
```

```
total = r + b
empty = 0

'This loop counts the number of allowable additions
FOR i = 1 TO 12
    IF soldat(i, 1) = 0 THEN empty = 1 + empty
NEXT i

'if too many additions are desired then
IF empty < total THEN
    LOCATE 8, 20: PRINT "You cannot add that many soldiers. "
    LOCATE 10, 20: PRINT "You must delete at least"; empty -
&   total; "soldiers first"
    LOCATE 12, 20: PRINT "HIT <CR> TO GO TO MAIN MENU"
    e$ = INPUT$(1)
    GOTO 10
END IF
IF r > 0 THEN CALL red(2, r)
IF b > 0 THEN CALL blue(2, b)
10 END SUB


SUB addevnt
'*************************************************************
'The subprogram ADDEVNT allows the user to add up to 25 events
'to the initial event list.  The program first checks to
'ensure that the user only intends to add less than 25 events,
'then calls EDITEVNT which displays the current event list
'and accepts input.
'*************************************************************
'VARIABLES: r = number of desired added events
'            total = current number of events on list

CLS
CALL frame(10, 75, 4, 8, 15, 9)
LOCATE 5, 20: PRINT " You can only add 25 events max."
LOCATE 6, 20: INPUT "How many events do you wish to add?"; r
total = r
'loop to count current events
FOR i = 1 TO 99
  IF event(i, 1) > 0 THEN total = total + 1
NEXT i

'If there are already 46 events on th  list, no more can be
'added
IF total > 46 THEN
    LOCATE 12, 20: PRINT "You cannot add that many events. "
    LOCATE 14, 20: PRINT "You must delete at least"; total -
&    46; "   events first"
    LOCATE 16, 20: PRINT "HIT <CR> TO GO TO MAIN MENU"
    e$ = INPUT$(1)
    GOTO 20
END IF
```

```
IF r > 0 THEN CALL editevnt(2, r)
20 END SUB


SUB addwpn
'****************************************************************
'The subprogram ADDWPN accepts user input accuracy data and
'creates phit tables.  Input data must be for ranges of 100-
'800 meters in increments of 100m.  It is assumed the data is
'in milliradians.  The user must also input the number of
'rounds per burst of fire.  The method of calculating the
'phit data is the Polya-Williams approximation.
'****************************************************************

'VARIABLES: n defines the weapon type.  The first user input
'               weapon is #7, the second is #8.

CLS
DIM a4(8, 5)  'user input accuracy data includes horizontal
'               and vertical aim and burst error
DIM sx(8)     'array of horizontal standard error
DIM sy(8)     'array of vertical standard error
DIM wid(8)    'array of P-W X factors
DIM tall(8, 3)' matrix of P-W Y factors
DIM pht(8, 3) 'Phit tables

n=7
OPEN "p4hit.exp" FOR OUTPUT AS #n

31 CALL frame(10, 70, 2, 5, 15, 9)'presentation screen
LOCATE 3, 20: PRINT "ADDING A NEW WEAPON TO SPARTAN"
LOCATE 7, 1: PRINT "To add a new weapon to SPARTAN, you must
&       have accuracy data"
PRINT " (aim error and ballistic error) for eight ranges.  If
&       you have less than"
PRINT "eight ranges, fill out the remainder with 0s."
LOCATE 11, 1: PRINT "input the requirod as the curser
%       indicates:"

'presentation screen for user input accuracy data
LOCATE 12, 1: PRINT "RANGE     AIM ERROR (mils)    BALLISTIC
&     ERROR (mils)"
LOCATE 13, 1: PRINT " (m)       HORIZ    VERT        HORIZ
&     VERT"

'This loop accepts accuracy data for each range
FOR i = 1 TO 8
     LOCATE 14 + i, 1, 1: PRINT USING "###m"; 100 * i:
&      a4(i, 1) = 100 * i               'range
     LOCATE 14 + i, 11, 1: INPUT a4(i, 2)'horizontal aim error
     LOCATE 14 + i, 21, 1: INPUT a4(i, 3)'vertical aim error
     LOCATE 14 + i, 33, 1: INPUT a4(i, 4)'horiz ballistic err
     LOCATE 14 + i, 42, 1: INPUT a4(i, 5)'vert ballistic err
```

```
NEXT i

'This loop computes the total standard error in the vertical
and horizontal directions

FOR i = 1 TO 8
  sx(i) = (a4(i, 2) * a4(i, 1) / 1000) ^ 2 + (a4(i, 4) *
&   a4(i, 1) / 1000) ^ 2
  sy(i) = (a4(i, 3) * a4(i, 1) / 1000) ^ 2 + (a4(i, 5) *
&   a4(i, 1) / 1000) ^ 2
'compute P-W X term
  wid(i) = 1 - EXP(-.32 / (3.14 * sx(i)))
'compute P-W Y term
  tall(i, 1) = 1 - EXP(-1.62 / (3.14 * sy(i,))
  tall(i, 2) = 1 - EXP(-.405 / (3.14 * sy(i)))
  tall(i, 3) = 1 - EXP(-.101 / (3.14 * sy(i)))
  p4(i, 1) = a4(i, 1)

  'This loop computes the phit data
  FOR j = 1 TO 3
      pht(i, j) = SQR(wid(i) * tall(i, j))
  NEXT j
NEXT i
PRINT "how many rounds in a burst?"
INPUT m!

'This loop uses the negative binomial function to compute the
'Phit for a burst weapon
FOR i = 1 TO 8
     FOR j = 2 TO 4
       p4(i, j) = 1 - (1 - pht(i, j - 1)) ^ m!
     NEXT j
NEXT i
CLS

'Print the user input weapon's Phit table
LOCATE 5, 1: PRINT "Here is your weapon's data."
LOCATE 10, 1: PRINT "RANGE           TARGET POSTURE"
LOCATE 11, 1: PRINT " (m)           STANDING     CROUCHING
PRONE"
LOCATE 14, 1
FOR i = 1 TO 8
PRINT p4(i, 1), p4(i, 2), p4(i, 3), p4(i, 4)  'print data
PRINT #n, p4(i, 2), p4(i, 3), p4(i, 4)    'write data to file
NEXT i
e$ = INPUT$(1)
CLOSE #n
'ask the user if he desires to input a second weapon type
IF n = 8 THEN GOTO 32
INPUT "Do you want to add another weapon type ? Y/N"; ans$
IF ans$ = "Y" OR ans$ = "y" THEN
 OPEN "p5hit.exp" FOR OUTPUT AS #7
 n = 8
 CLS
 GOTO 31
 END IF
32 END SUB


SUB blue (opt%, b)
'*************************************************************
'Subprogram BLUE displays the current BLUE soldier data and
'then, depending on the user option, will edit, add or delete
'a soldier.
'*************************************************************
'VARIABLES: opt% is the variable that declares what option the
'            subprogram executes
'            b is the number of soldiers to be edited

CLS
WIDTH 80, 25

'This presentation screen displays the current BLUE data
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 30: PRINT "BLUE SOLDIER DATA"
LOCATE 8, 1: PRINT "SOLDIER     DUTY       LOCATION   MOVEMENT
    STATUS    POSTURE    WEAPON"
LOCATE 9, 1: PRINT "          POSITION               DIRECTION"
FOR i = 1 TO 12
 IF soldat(i, 1) < 1 THEN GOTO 65
 LOCATE 10 + i, 3: PRINT i
 LOCATE 10 + i, 11: ON soldat(i, 2) GOTO 41, 42, 43, 44, 45
41 PRINT "SQD LDR"                          'duty postion
   GOTO 46
42   PRINT "TEAM LDR"
   GOTO 46
43   PRINT "GRENADIER"
   GOTO 46
44 PRINT "SAW GUNNER"
 GOTO 46
45 PRINT "RIFLEMAN"
46 LOCATE 10 + i, 23: PRINT USING "###"; soldat(i, 3);
soldat(i, 4)
LOCATE 10 + i, 36

'This next algorithm converts radians into the map's 360
'degree data
dir = soldat(i, 8)
 dir = 90 - dir * 180 / 3.141
 IF dir < 0 THEN dir = 360 + dir

PRINT USING "###"; dir
LOCATE 10 + i, 44:
IF soldat(i, 9) = 0 THEN      'move status
PRINT "STATIONARY"
```

```
GOTO 50
END IF
PRINT "MOVING"
50 LOCATE 10 + i, 57: ON soldat(i, 10) GOTO 51, 52, 53
51 PRINT "STANDING"            'posture
GOTO 54
52 PRINT "CROUCHING"
GOTO 54
53 PRINT "PRONE"
54 LOCATE 10 + i, 69: ON soldat(i, 11) GOTO 61, 62, 63, 64
61 IF soldat(i, 2) = 3 THEN              'weapon type
PRINT "M203"
GOTO 65
END IF
PRINT "M16A2"
GOTO 65
62 PRINT "AK-74"
GOTO 65
63 PRINT "SAW"
GOTO 65
64 PRINT "OTHER"
65 NEXT i
LOCATE 1, 25: PRINT "HIT <CR> to continue"
e$ = INPUT$(1)

'opt%=0 means no editing is desired
IF opt% = 0 THEN GOTO 79

'opt%=2 means to add additional soldiers
IF opt% = 2 THEN GOTO 70

'This portion deletes BLUE soldiers
66 LOCATE 11 + i: INPUT "Which soldier do you want to
&   delete?", d%
IF soldat(d%, 1) < 1 THEN
    PRINT "The number you input is not a blue soldier, try
&     again."
    GOTO 66
END IF

'This loop zeros out all soldier data
FOR i = 1 TO 15
     soldat(d%, i) = 0
NEXT i
GOTO 79'   goto the end of the subprogram

'Input soldier data
70 CLS
LOCATE 2, 20: PRINT "INPUT BLUE SOLDIER DATA"
LOCATE 6, 4: PRINT "DUTY POSITION"
LOCATE 7, 4: PRINT "X GRID COORD"
LOCATE 8, 4: PRINT "Y GRID COORD"
LOCATE 9, 4: PRINT "#GRENADES"
LOCATE 10, 4: PRINT "TIME FIRED"
LOCATE 11, 4: PRINT "NOT USED"
LOCATE 12, 4: PRINT "MOVEMENT DIRECTION"
LOCATE 13, 4: PRINT "MOVEMENT STATUS"
LOCATE 14, 4: PRINT "POSTURE"
LOCATE 15, 4: PRINT "WEAPON TYPE"
LOCATE 16, 4: PRINT "ROUNDS PER MAGAZINE"
LOCATE 17, 4: PRINT "NUMBER MAGAZINES"
LOCATE 18, 4: PRINT "TARGET ID"
LOCATE 19, 4: PRINT "WOUND STATUS"
FOR i = 1 TO r
    FOR j = 1 TO 12
       IF soldat(j, 2) > 0 THEN GOTO 71
       LOCATE 3, 18 + i * 10: PRINT "SOLDIER"; i
       soldat(j, 1) = 1
       FOR k = 2 TO 15
           LOCATE 3 + k, 18 + i * 10: INPUT soldat(j, k)
       NEXT k
       GOTO 72
    71 NEXT j
72 NEXT i
79 END SUB



SUB contour
'*************************************************************
'Subprogram CONTOUR prints the map screen, then, in increments
'of 10 meters, highlights all terrain cells having elevations
'contained by the contour intervals.
'*************************************************************

'copies the map on the nondisplayed screen to the visible
'screen
PCOPY 1, 0

'This loop starts at the lowest elevation level in SPARTAN and
'highlights the center of each terrain cell having a elevation
'batween the two levels
FOR k = 0 TO 40 STEP 10
   LOCATE 1, 1: PRINT "Elevation between"; 60 + k; "and";
&     69 + k; "meters"
   FOR i = 1 TO 50
     FOR j = 1 TO 50
'    if the elevation of the cell is witin the current
'    interval, highlight the center of the cell
     IF mapl(i, j, 3) > 59 + k AND mapl(i, j, 3) < 70 + k THEN
        PSET (i * 20 - 10, j * 20 - 10), 0
     END IF
     NEXT j
   NEXT i
LOCATE 2, 1: PRINT "HIT <CR> to continue"
```

```
e$ = INPUT$(1)

'copy the undisplayed map to the current screen to do the next
'contour interval
   PCOPY 1, 0
NEXT k
END SUB



SUB cphit
'*****************************************************************
'The subprogram cphit displays the raw accuracy data for the
'default weapons and then displays the computed phit tables
'*****************************************************************

CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: PRINT "CURRENT ACCURACY DATA FOR M16A2"
LOCATE 10, 1: PRINT "RANGE     AIM ERROR (mils)     BALLISTIC
ERROR (mils)"
LOCATE 11, 1: PRINT " (m)          HORIZ     VERT          HORIZ
  VERT"
PRINT " 100          4.7       4.7          7.85      8.98  "
PRINT " 200          3.5       3.5          7.85      8.98"
PRINT " 300          3.1       3.1          7.85      8.98"
PRINT " 400          2.9       2.9          7.85      8.98"
PRINT " 500          2.8       2.8          7.85      8.98  "
PRINT " 600          2.7       2.7          7.85      8.98"
PRINT " 700          2.6       2.6          7.85      8.98"
PRINT " 800          2.5       2.5          7.85      8.98'
LOCATE 23, 15: PRINT "HIT <CR> TO SEE P(HIT)"
e$ = INPUT$(1)
CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: PRINT "CURRENT P(hit) DATA FOR M16A2"
LOCATE 10, 1: PRINT "RANGE                          TARGET POSTURE"
LOCATE 11, 1: PRINT " (m)               STANDING      CROUCHING
  PRONE"
LOCATE 14, 1
FOR i = 1 TO 8
   PRINT pl(i, 1)  pl(i, 2), pl(i, 3), pl(i, 4)
NEXT i

'ask the user if he desires to view more data
LOCATE 23, 15: PRINT "Do you want to see other P(hit) tables?
&   Y/N"
ans$ = INPUT$(1)
IF ans$ = "Y" OR ans$ = "y" THEN GOTO 81
GOTO 80

'Accuracy data for the AK74
81 CLS
```

```
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: PRINT "CURRENT ACCURACY DATA FOR AK-74"
LOCATE 10, 1: PRINT "RANGE      AIM ERROR (mils)     BALLISTIC
ERROR (mils)"
LOCATE 11, 1: PRINT " (m)          HORIZ     VERT           HORIZ
  VERT"
PRINT " 100          4.7       4.7            8.47        8.9"
PRINT " 200          3.5       3.5            8.47        8.9"
PRINT " 300          3.1       3.1            8.47        8.9"
PRINT " 400          2.9       2.9            8.47        8.9  "
PRINT " 500          2.8       2.8            8.47        8.9"
PRINT " 600          2.7       2.7            8.47        8.9"
PRINT " 700          2.6       2.6            8.47        8.9"
PRINT " 800          2.6       2.6            8.47        8.9"
LOCATE 23, 15: PRINT "HIT <CR> TO SEE P(HIT)"
e$ = INPUT$(1)
CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: PRINT "CURRENT P(hit) DATA FOR AK-74"
LOCATE 10, 1: PRINT "RANGE                        TARGET POSTURE"
LOCATE 11, 1: PRINT " (m)                  STANDING      CROUCHING
  PRONE"
LOCATE 14, 1
FOR i = 1 TO 8
PRINT p2(i, 1), p2(i, 2), p2(i, 3), p2(i, 4)
NEXT i
LOCATE 23, 15: PRINT "Do you want to see other P(hit) tables?
Y/N"
ans$ = INPUT$(1)
IF ans$ = "Y" OR ans$ = "y" THEN GOTO 82
GOTO 80

'Accuracy data for the SAW and RPK74
82 CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: PRINT "CURRENT ACCURACY DATA FOR SAW/RPK-74"
LOCATE 10, 1: PRINT "RANGE      AIM ERROR (mils)     BALLISTIC
ERROR (mils)"
LOCATE 11, 1: PRINT " (m)          HORIZ     VERT           HORIZ
  VERT"
PRINT "100          6.1       6.1            1.27        1.41  "
PRINT "200          4.1       4.1            1.27        1.41"
PRINT "300          3.1       3.1            1.27        1.41"
PRINT "600          2.7       2.7            1.27        1.41"
PRINT "900          2.5       2.5            1.27        1.41"
PRINT "1000         2.5       2.5            1.27        1.41"
PRINT "1800         2.4       2.4            1.27        1.41"
PRINT "2100         2.4       2.4            1.27        1.41"
LOCATE 23, 15: PRINT "HIT <CR> TO SEE P(HIT)"
e$ = INPUT$(1)
CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: PRINT "CURRENT P(hit) DATA FOR SAW/RPK74"
LOCATE 10, 1: PRINT "RANGE                    TARGET POSTURE"
LOCATE 11, 1: PRINT " (m)              STANDING     CROUCHING
&    PRONE"
LOCATE 14, 1
FOR i = 1 TO 8
    PRINT p3(i, 1), p3(i, 2), p3(i, 3), p3(i, 4)
NEXT i
LOCATE 23, 15: PRINT "HIT <CR> to return to menu"
e$ = INPUT$(1)
80 END SUB


SUB default
'***************************************************************
'The subprogram default takes all default data files and
'creates the .exp files that are read by SPARTAN for
'simulation execution.  It also creates empty obstacle and
'user input phit files
'***************************************************************
CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 22: PRINT "STANDBY WHILE DEFAULT DATA FILES LOAD"

OPEN "mapl.dat" FOR INPUT AS #1      'terrain data file
OPEN "mapl.exp" FOR OUTPUT AS #2
FOR i = 1 TO 50
FOR j = 1 TO 50
 INPUT #1, mapl(i, j, 1), mapl(i, j, 2), mapl(i, j, 3)
 PRINT #2, mapl(i, j, 1), mapl(i, j, 2), mapl(i, j, 3)
 NEXT j
 NEXT i
CLOSE #1
CLOSE #2

OPEN "m16.dat" FOR INPUT AS #1      'M16 Phit table
OPEN "ak74.dat" FOR INPUT AS #2     'AK74 Phit table
OPEN "saw.dat" FOR INPUT AS #3      'SAW Phit table
OPEN "P1HIT.exp" FOR OUTPUT AS #4
OPEN "P2HIT.exp" FOR OUTPUT AS #5
OPEN "P3HIT.exp" FOR OUTPUT AS #6
FOR i = 1 TO 8
FOR j = 1 TO 4
 INPUT #1, p1(i, j)
 INPUT #2, p2(i, j)
 INPUT #3, p3(i, j)
 NEXT j
 NEXT i
FOR i = 1 TO 8
FOR j = 2 TO 4
 PRINT #4, p1(i, j)
 PRINT #5, p2(i, j)
 PRINT #6, p3(i, j)
 NEXT j
 NEXT i
 CLOSE #1: CLOSE #2: CLOSE #3: CLOSE #4: CLOSE #5: CLOSE #6

OPEN "P4HIT.exp" FOR OUTPUT AS #1    'user input Phit data
OPEN "P5HIT.exp" FOR OUTPUT AS #2    'user input Phit data
CLOSE #1: CLOSE #2
OPEN "obs.exp" FOR OUTPUT AS #1
CLOSE #1

OPEN "event.dat" FOR INPUT AS #1     'initial event list
OPEN "event.exp" FOR OUTPUT AS #2
FOR i = 1 TO 24
FOR j = 1 TO 3
 INPUT #1, event(i, j)
 PRINT #2, event(i, j)
 NEXT j
 NEXT i
CLOSE #1: CLOSE #2

OPEN "joe.dat" FOR INPUT AS #1      'soldier attribute list
OPEN "joe.exp" FOR OUTPUT AS #2
FOR i = 1 TO 12
FOR j = 1 TO 15
 INPUT #1, soldat(i, j)
 PRINT #2, soldat(i, j)
NEXT j
NEXT i
CLOSE #1: CLOSE #2
END SUB


SUB delete
'************************************************************
'The subprogram DELETE checks to see what type of soldier the
'user wishes to delete, then calls to either RED or BLUE to
'delete the soldier's data.
'************************************************************

CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: INPUT "Do you wish to delete RED or BLUE
soldiers? R/B"; ans$
IF ans$ = "R" OR ans$ = "r" THEN
CALL red(1, 0)
GOTO 90
END IF
CALL blue(1, 0)
90 END SUB
```

```
SUB delevnt
'*************************************************************
'This subprogram allows the user to delete events from the
'initial event file.  After inputing the number of events to
'be deleted, the subprogram call to edit event to delete the
'events.
'*************************************************************

CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: INPUT "How many events do you wish to delete";d
IF d > 0 THEN
CALL editevnt(1, d)
GOTO 100
END IF
100 END SUB


SUB editevnt (opt, num)
'*************************************************************
'EDITEVNT allows the user to view the current event list and
'then edit, add or delete events
'*************************************************************
'VARIABLES:  k= number of events currently displayed on the
'screen.  This variable is used to control the screen display
'            opt = the user option for file editing
'                 1 = delete, 2 = add, 3 = edit, 4=view

CLS
WIDTH 80, 25
CALL frame(10, 70, 4, 7, 15, 9)

'first, display the current event list

LOCATE 5, 30: PRINT "CURRENT EVENT LIST"
LOCATE 8, 1: PRINT "EVENT   TYPE EVENT              ACTOR
  TIME SCHEDULED"
k = 0
VIEW PRINT 10 TO 24
FOR i = 1 TO 99
 IF event(i, 1) = 0 THEN GOTO 110
 k = k + 1
 IF k < 10 THEN GOTO 111
 LOCATE 12 + k: PRINT "Hit <CR> to continue"
e$ = INPUT$(1)
CLS
k = 0
111 LOCATE 10 + k, 3: PRINT i

'translate the event type into verbage

 LOCATE 10 + k, 11: ON event(i, 1) GOTO 112, 113, 114, 115,
116, 146
112 PRINT "Search"
   GOTO 117
113   PRINT "Select tgt"
   GOTO 117
114   PRINT "Engage tgt"
   GOTO 117
115 PRINT "Move"
   GOTO 117
116 PRINT "React to fire"
   GOTO 117
146 PRINT "Indirect fire"
117 LOCATE 10 + k, 30: PRINT USING "###.##"; event(i, 2)
LOCATE 10 + k, 46: PRINT USING "###.##"; event(i, 3)
110 NEXT i
VIEW PRINT
LOCATE 1, 25: PRINT "HIT <CR> to continue"
e$ = INPUT$(1)
CLS

'opt is the option for editing

ON opt GOTO 120, 121, 122, 130

' delete one or more events
120 CLS
LOCATE 3: INPUT "Which event do you want to delete?", e%
FOR i = 1 TO 99
   IF event(i, 1) = 0 THEN GOTO 123
     k = k + 1
123 NEXT i
 IF e% > k THEN
CLS
PRINT "The number you input is not a scheduled event, try
&    again."
GOTO 120
END IF

'loop zeros out event on list
FOR i = 1 TO 3
  event(e%, i) = 0
NEXT i

INPUT "Do you wish to delete any more scheduled events? Y/N";
ans$
IF ans$ = "Y" OR ans$ = "y" THEN GOTO 120
CLS
GOTO 130 'return to the previous subprogram

'add one or more events
121 CLS
FOR i = 1 TO 22
    LOCATE 8, 10: PRINT "TYPE EVENT       ACTOR       TIME
```

```
&     SCHEDULED"
     k = 0
     IF event(i, 1) > 0 GOTO 124
     k = k + 1
     LOCATE 10 + k, 1: PRINT "Input"
     LOCATE 10 + k, 15: INPUT event(i, 1)
     LOCATE 10 + k, 28: INPUT event(i, 2)
     LOCATE 10 + k, 40: INPUT event(i, 3)
     LOCATE 23, 1: INPUT "Do you wish to add any more
&     events?Y/N"; ans$
IF ans$ = "Y" OR ans$ = "y" THEN GOTO 124
GOTO 130'  return to the previous subprogram

   124 CLS
NEXT i
GOTO 130 'return to the previous subprogram, no more events
         '    can be added

'Edit an event currently on the list
122 CLS
LOCATE 11 + k: INPUT "Which event do you wish to edit?"; e
CLS
LOCATE 5, 30: PRINT "Here is the current event"
LOCATE 8, 1: PRINT "EVENT   TYPE EVENT       ACTOR       TIME
SCHEDULED"
 LOCATE 10, 3: PRINT e
 LOCATE 10, 11: ON event(e, 1) GOTO 125, 126, 127, 128, 129,
147
125 PRINT "Search (1)"
    GOTO 131
126 PRINT "Select tgt (2)"
    GOTO 131
127 PRINT "Engage tgt (3)"
    GOTO 131
128 PRINT "Move (4)"
    GOTO 131
129 PRINT "React to fire (5)"
    GOTO 131
147 PRINT "Indirect fire (6)"

131 LOCATE 10, 25: PRINT USING "###.##"; event(e, 2)
LOCATE 10, 40: PRINT USING "###.##"; event(e, 3)
LOCATE 11, 1: PRINT "Input"
LOCATE 11, 15: INPUT event(e, 1)
LOCATE 11, 25: INPUT event(e, 2)
LOCATE 11, 40: INPUT event(e, 3)
LOCATE 13, 1: INPUT "Do you wish to edit any more events?
Y/N"; ans$
CLS
IF ans$ = "Y" OR ans$ = "y" THEN GOTO 122
130 END SUB
```

```
SUB elist
'**************************************************************
'ELIST is the Event List Editor Menu.  It downloads the
'default data file, allows the user to access different editor
'functions and creates the .exp event file for SPARTAN.
'**************************************************************

CLS
COLOR 15, 9
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 20: PRINT "STANDBY WHILE EVENT FILE IS DOWNLOADED"
OPEN "event.dat" FOR INPUT AS #1
FOR i = 1 TO 24
   FOR j = 1 TO 3
     INPUT #1, event(i, j)
   NEXT j
NEXT i
CLOSE #1
DO
CLS
left% = 10: right% = 70: top% = 4: bottom% = 20: fore% = 15:
back% = 9
CALL frame(left%, right%, top%, bottom%, fore%, back%)
LOCATE 6, 20: PRINT "EVENT DATA FILE EDITOR MENU"
LOCATE 8, 20: PRINT "1) Look at scheduled events"
LOCATE 10, 20: PRINT "2) Add event"
LOCATE 12, 20: PRINT "3) Delete event"
LOCATE 14, 20: PRINT "4) Edit event list"
LOCATE 16, 20: PRINT "5) Event Editor Help"
LOCATE 18, 20: PRINT "6) Exit to Main Menu"
LOCATE 19, 20: ch$ = INPUT$(1)
SELECT CASE ch$
CASE "1"
  CALL editevnt(4, 0)
CASE "2"
  CALL addevnt
CASE "3"
 CALL delevnt
 CASE "4"
 CALL editevnt(3, 0)
 CASE "5"
 CALL evnthelp
 CASE "6"
'Case 6 is to exit the Event Editor.  Before exiting, the
altered data is loaded into the .exp file for SPARTAN

OPEN "event.exp" FOR OUTPUT AS #2
FOR i = 1 TO 99
     FOR j = 1 TO 3
       PRINT #2, event(i, j)
     NEXT j
NEXT i
CLOSE #2
```

```
 EXIT DO
CASE ELSE
  BEEP
   LOCATE 18, 20: PRINT "Try again, GOMER"
   ch$ = INPUT$(1)
END SELECT
LOOP
END SUB


SUB explain
'**************************************************************
'EXPLAIN presents information for preprocessor users for
'running STARTUP
'**************************************************************

CLS
COLOR 15, 1
CALL frame(10, 70, 4, 7, 15, 1)
LOCATE 5, 20: PRINT "WELCOME TO SPARTAN STARTUP PROGRAM"
LOCATE 9, 4: PRINT "The purpose of this program is to review
and or modify SPARTAN's"
LOCATE 10, 4: PRINT "default data files."
LOCATE 11, 4: PRINT "If you have any questions about SPARTAN
processes during the simulation"
LOCATE 12, 4: PRINT "run refer to the USER GUIDE or hit <F1>,
that will bring up the"
LOCATE 13, 4: PRINT "HELP MENU."
LOCATE 24, 1: INPUT "Hit <CR> to continue to the main menu.",
ans$
END SUB


SUB format
'**************************************************************
'FORMAT allows the user to pick a BLUE formation and squad
'leader location.  It then updates all the rest of the squad
'member's locations.
'**************************************************************

'VARIABLES:  array is the data array that contains the
'information neccessary to draw the icons in the formation
'diagrams

SCREEN 9
WIDTH 80, 43
DIM array(0 TO 104) AS INTEGER

'Establish four view ports on the screen, and diagram the
'formations
```

```
VIEW (39, 0)-(299, 125), 9, 0
LOCATE 1, 10: PRINT "DIRECTION OF MOVEMENT"; CHR$(24)

'draw stick man
LINE (133, 55)-(135, 57), 15, BF
LINE (134, 55)-(134, 61), 15
LINE (134, 62)-(137, 68), 15
LINE (134, 62)-(131, 68), 15
LINE (135, 59)-(138, 62), 15
LINE (133, 59)-(130, 62), 15
'put man into data array
GET (130, 55)-(138, 68), array

' draw first formation

PUT (180, 70), array, PSET
PUT (230, 70), array, PSET
PUT (80, 70), array, PSET
PUT (30, 70), array, PSET
PUT (65, 30), array, PSET
PUT (175, 30), array, PSET
PUT (150, 90), array, PSET
PUT (110, 90), array, PSET
LOCATE 16, 10: PRINT "1. SQUAD LINE/TEAM WEDGE"
VIEW

'     2nd formation

VIEW (341, 0)-(600, 125), 9, 0
LOCATE 1, 47: PRINT "DIRECTION OF MOVEMENT"; CHR$(26)
PUT (144, 60), array, PSET
PUT (240, 60), array, PSET
PUT (90, 60), array, PSET
PUT (210, 40), array, PSET
PUT (210, 80), array, PSET
PUT (180, 100), array, PSET
PUT (60, 40), array, PSET
PUT (60, 80), array, PSET
PUT (30, 20), array, PSET
LOCATE 16, 47: PRINT "2. SQUAD COLUMN/TEAM WEDGE"

VIEW
'    3rd formation

VIEW (39, 175)-(299, 300), 9, 0
LOCATE 23, 10: PRINT "DIRECTION OF MOVEMENT"; CHR$(26)
PUT (140, 60), array, PSET
PUT (240, 60), array, PSET
PUT (90, 60), array, PSET
PUT (210, 40), array, PSET
PUT (210, 80), array, PSET
PUT (180, 60), array, PSET
PUT (60, 40), array, PSET
PUT (60, 80), array, PSET
PUT (30, 60), array, PSET
LOCATE 38, 10: PRINT "3. SQUAD COLUMN/TEAM WEDGE"

VIEW
'      4th formation

VIEW (341, 175)-(600, 300), 9, 0
LOCATE 23, 47: PRINT "DIRECTION OF MOVEMENT"; CHR$(26)
PUT (250, 60), array, PSET
PUT (225, 60), array, PSET
PUT (200, 60), array, PSET
PUT (170, 60), array, PSET
PUT (140, 60), array, PSET
PUT (110, 60), array, PSET
PUT (80, 60), array, PSET
PUT (50, 60), array, PSET
PUT (20, 60), array, PSET
LOCATE 38, 47: PRINT "4. SQUAD FILE"
VIEW

' query the user for his formation and location choices

140 LOCATE 40, 1: INPUT "Input formation number <e.g. 3>", n%
 LOCATE 41, 1: INPUT "Input direction of movement in degrees
&    <e.g. 45>", dir
LOCATE 42, 1: INPUT "Input squad leader x and y grid
&   coordinates <e.g.200,200>", x, y
x1 = x - 8: y1 = y - 10

'convert the user input direction from degrees to radians

      dir = (90 - dir) * 3.141 / 180

'update soldier's location attributes based on user choice

ON n% GOTO 141, 142, 143, 144
PRINT "TRY AGAIN"
GOTO 140

'                  1st formation
141 CLS
soldat(1, 3) = x: soldat(1, 4) = y
soldat(5, 3) = x + 36 * COS(2.55 + dir): soldat(5, 4) = y + 36
&  * SIN(2.55 + dir)
soldat(3, 3) = x + 30 * COS(1.57 + dir): soldat(3, 4) = y + 30
&  * SIN(1.57 + dir)
soldat(2, 3) = x + 56 * COS(1.39 + dir): soldat(2, 4) = y + 56
&  * SIN(1.39 + dir)
soldat(4, 3) = x + 65.7 * COS(1.72 + dir): soldat(4, 4) = y +
&  65.7 * SIN(1.72 + dir)
soldat(9, 3) = x + 36 * COS(-2.55 + dir): soldat(9, 4) = y +
&  36 * SIN(-2.55 + dir)
```

```
soldat(7, 3) = x + 30 * COS(-1.57 + dir): soldat(7, 4) = y +
&  30 * SIN(-1.57 + dir)
soldat(6, 3) = x + 56 * COS(-1.39 + dir): soldat(6, 4) = y +
&  56 * SIN(-1.39 + dir)
soldat(8, 3) = x + 65.7 * COS(-1.72 + dir): soldat(8, 4) = y
&  + 65.7 * SIN(-1.72 + dir)
GOTO 145

'             2nd formation
142 CLS
soldat(1, 3) = x: soldat(1, 4) = y
soldat(2, 3) = x + 75 * COS(dir): soldat(2, 4) = y + 75 *
&    SIN(dir)
soldat(3, 3) = x + 56 * COS(-.464 + dir): soldat(3, 4) = y +
&   56 * SIN(-.464 + dir)
soldat(4, 3) = x + 56 * COS(.464 + dir): soldat(4, 4) = y + 56
&   * SIN(.464 + dir)
soldat(5, 3) = x + 56 * COS(-1.11 + dir): soldat(5, 4) = y +
&   56 * SIN(-1.11 + dir)
soldat(6, 3) = x + 25 * COS(3.141 + dir): soldat(6, 4) = y +
&   25 * SIN(3.141 + dir)
soldat(7, 3) = x + 56 * COS(3.6 + dir): soldat(7, 4) = y + 56
&   * SIN(3.6 + dir)
soldat(8, 3) = x + 56 * COS(-3.6 + dir): soldat(8, 4) = y + 56
&   * SIN(-3.6 + dir)
soldat(9, 3) = x + 90 * COS(2.55 + dir): soldat(9, 4) = y + 90
&   * SIN(2.55 + dir)
GOTO 145

'              3rd choice
143 CLS
soldat(1, 3) = x: soldat(1, 4) = y
soldat(2, 3) = x + 95 * COS(dir): soldat(2, 4) = y + 95 *
&  SIN(dir)
soldat(3, 3) = x + 70 * COS(-.464 + dir): soldat(3, 4) = y +
&  70 * SIN(-.464 + dir)
soldat(4, 3) = x + 70 * COS(.464 + dir): soldat(4, 4) = y + 70
&  * SIN(.464 + dir)
soldat(5, 3) = x + 40 * COS(dir): soldat(5, 4) = y + 40 *
&  SIN(dir)
soldat(6, 3) = x + 30 * COS(3.141 + dir): soldat(6, 4) = y +
&  30 * SIN(3.141 + dir)
soldat(7, 3) = x + 56 * COS(3.6 + dir): soldat(7, 4) = y + 56
&  * SIN(3.6 + dir)
soldat(8, 3) = x + 56 * COS(-3.6 + dir): soldat(8, 4) = y + 56
&  * SIN(-3.6 + dir)
soldat(9, 3) = x + 75 * COS(3.141 + dir): soldat(9, 4) = y +
&  75 * SIN(3.141 + dir)
GOTO 145


'              4th formation
```

```
144 CLS
soldat(1, 3) = x: soldat(1, 4) = y
soldat(2, 3) = x + 100 * COS(dir): soldat(2, 4) = y + 100 *
& SIN(dir)
soldat(3, 3) = x + 75 * COS(dir): soldat(3, 4) = y + 75 *
& SIN(dir)
soldat(4, 3) = x + 50 * COS(dir): soldat(4, 4) = y + 50 *
& SIN(dir)
soldat(5, 3) = x + 25 * COS(dir): soldat(5, 4) = y + 25 *
& SIN(dir)
soldat(6, 3) = x + 25 * COS(3.141 + dir): soldat(6, 4) = y +
& 25 * SIN(3.141 + dir)
soldat(7, 3) = x + 50 * COS(3.141 + dir): soldat(7, 4) = y +
& 50 * SIN(3.141 + dir)
soldat(8, 3) = x + 75 * COS(3.141 + dir): soldat(8, 4) = y +
& 75 * SIN(3.141 + dir)
soldat(9, 3) = x + 100 * COS(3.141 + dir): soldat(9, 4) = y +
& 100 * SIN(3.141 + dir)

'update all soldier's location to reflect user input direction
145 FOR i = 1 TO 9
  soldat(i, 8) = dir
NEXT i
END SUB


SUB frame (left%, right%, top%, bottom%, fore%, back%)
'*************************************************************
'FRAME draws the frames seen on presentation screens
'*************************************************************

COLOR fore%, back%
 LOCATE top%, left%: PRINT CHR$(201)
LOCATE top%, right%: PRINT CHR$(187)
LOCATE bottom%, left%: PRINT CHR$(200)
LOCATE bottom%, right%: PRINT CHR$(188)
  FOR vert% = top% + 1 TO bottom% - 1
    LOCATE vert%, left%: PRINT CHR$(186)
    LOCATE vert%, right%: PRINT CHR$(186)
  NEXT vert%
horiz% = right% - left% - 1
hline$ = STRING$(horiz%, 205)
LOCATE top%, left% + 1: PRINT hline$
LOCATE bottom%, left% + 1: PRINT hline$
END SUB
```

```
SUB help
'*****************************************************************
'Subprogram HELP is the main menu for the help screens.  It
'queries the user for the specific help function the user
'desires, then accesses that file
'*****************************************************************

DO
CLS
left% = 10: right% = 70: top% = 4: bottom% = 23: fore% = 15:
back% = 1
WIDTH 80, 25
CALL frame(left%, right%, top%, bottom%, fore%, back%)
LOCATE 6, 30: PRINT " HELP MENU"
LOCATE 8, 20: PRINT "1) Help with terrain editor"
LOCATE 10, 20: PRINT "2) Help with soldier attribute editor"
LOCATE 12, 20: PRINT "3) Help with Pr(hit) editor"
LOCATE 14, 20: PRINT "4) Help with event editor"
LOCATE 16, 20: PRINT "5) About SPARTAN"
LOCATE 18, 20: PRINT "6) Exit to main menu"
4 LOCATE 19, 20: ch$ = INPUT$(1)
SELECT CASE ch$
CASE "1"
  CALL maphelp
CASE "2"
  CALL joehelp
CASE "3"
  CALL phithelp
CASE "4"
  CALL evnthelp
CASE "5"
CALL aboutspartan
CASE "6"
EXIT DO
CASE ELSE
  BEEP
   LOCATE 20, 20: PRINT "Try again, GOMER"
GOTO 4
END SELECT
LOOP
END SUB


SUB joeatrib
'*****************************************************************
'JOEATRIB allows the user to edit a soldier's current
attribute 'values.
'*****************************************************************

150 CLS
CALL frame(10, 70, 1, 7, 15, 9)
LOCATE 2, 15: INPUT "Which side do you want to edit? R/B",
ans$
IF ans$ = "B" OR ans$ = "b" THEN
strng$ = "BLUE"
GOTO 151
END IF
strng$ = "RED"
151 LOCATE 4, 25: PRINT strng$; "  SOLDIER ATTRIBUTES"

'queries the user to ensure edited soldeir is a soldier

152 LOCATE 6, 21: INPUT "Which soldier do you wish to edit?";
ans
IF ans > 12 THEN
LOCATE 23, 4: PRINT "TRY another soldier,soldier"
GOTO 152
END IF

'display current values of soldier's attributes

CLS
LOCATE 2, 20: PRINT "OLD ATTRIBUTES"
LOCATE 2, 35: PRINT "NEW ATTRIBUTES"
LOCATE 6, 4: PRINT "SOLDIER #"
LOCATE 7, 4: PRINT "SIDE"
LOCATE 8, 4: PRINT "DUTY POSITION"
LOCATE 9, 4: PRINT "X GRID COORD"
LOCATE 10, 4: PRINT "Y GRID COORD"
LOCATE 11, 4: PRINT "#GRENADES"
LOCATE 12, 4: PRINT "TIME FIRED"
LOCATE 13, 4: PRINT "NOT USED"
LOCATE 14, 4: PRINT "MOVEMENT DIRECTION"
LOCATE 15, 4: PRINT "MOVEMENT STATUS"
LOCATE 16, 4: PRINT "POSTURE"
LOCATE 17, 4: PRINT "WEAPON TYPE"
LOCATE 18, 4: PRINT "ROUNDS PER MAGAZINE"
LOCATE 19, 4: PRINT "NUMBER MAGAZINES"
LOCATE 20, 4: PRINT "TARGET ID"
LOCATE 21, 4: PRINT "WOUND STATUS"
LOCATE 6, 15: PRINT ans
FOR i = 1 TO 15
   LCCATE 6 + i, 25: PRINT soldat(ans, i)

   'input new attribute values
   LOCATE 6 + i, 40: INPUT ; soldat(ans, i)
NEXT i
LOCATE 23, 1
INPUT "Do you wish to alter anymore soldiers? Y/N"; ans$
IF ans$ = "Y" OR ans$ = "y" GOTO 150
END SUB


SUB los
```

```
'**********************************************************
'LOS draws the line of sight cone for user input location,
'range, and stop and start angles.
'**********************************************************

'copy the map from the undisplayed screen to the visible one

160 PCOPY 1, 0

'query the user for input

LOCATE 1, 1: INPUT "Input observer's location. (Input X,Y)
&  (example <900,900> )"; x, y
LOCATE 2, 1: INPUT "Line of sight cone(start degree,end
&  degree) (exapmle <45,275>)"; start, fin
LOCATE 3, 1: INPUT "Line of sight radius (between 0 and 1000
&  meters)"; r

'convert start and finish angles from degrees to radians

 IF start >= 0 AND start <= 90 THEN
    start = (90 - start) * 3.141 / 180
 ELSE
    start = (360 - (start - 90)) * 3.141 / 180
 END IF

 IF fin >= 0 AND fin <= 90 THEN
    fin = (90 - fin) * 3.141 / 180
 ELSE
    fin = (360 - (fin - 90)) * 3.141 / 180
END IF

PCOPY 1, 0   'refresh the screen

'draw the line of sight cone
CIRCLE (x, y), r, 0, -fin, -start, 340 / 650

'determine observer elevation

hexx = INT(x / 20 + 1): hexy = INT(y / 20 + 1)
z1 = mapl(hexx, hexy, 3) + 1.8

'if observer is in the woods, assign the woods flag wl=1
IF mapl(hexx, hexy, 2) < 1 THEN
      wl = 1
  ELSE
      wl = 0
END IF

w = wl

'loop to check los every .05 radians around line of sight cone
FOR k = fin TO start STEP .05
```

```
'determine the target location (x2,y2)
x2 = x + r * COS(k)
IF x2 < 1 THEN x2 = 1
IF x2 >= 1000 THEN x2 = 999
y2 = y + r * SIN(k)
IF y2 < 1 THEN y2 = 1
IF y2 >= 1000 THEN y2 = 999

'determine target elevation
z2 = .9 + mapl(INT(x2 / 20 + 1), INT(y2 / 20 + 1), 3)

'determine slope of observer-target line
slope = (z2 - z1) / r

'loop every meter on the redius from observer to target

   FOR i = 1 TO r
'if line is off the map go to next angle
     xn = x + i * COS(k): yn = y + i * SIN(k)
     IF 0 >= xn OR xn >= 1000 THEN GOTO 166
     IF 0 >= yn OR yn >= 1000 THEN GOTO 166
'if the eleveation sheck is in the same terrain cell, go to
'the next meter along the radius
     IF INT(xn / 20 + 1) = hexx AND INT(yn / 20 + 1) = hexy
 &       THEN GOTO 165

'else compute the new elevation of the intervening terrain
'cell
      hexx = INT(xn / 20 + 1): hexy = INT(yn / 20 + 1)
      IF w = 1 AND mapl(hexx, hexy, 2) = 1 THEN w = 0

'if the cell is iin the woods and the observer is out of the
'woods factor in the elevation of the trees

      IF w = 0 AND mapl(hexx, hexy, 2) < 1 THEN
        znow = mapl(hexx, hexy, 3) + 10 / mapl(hexx, hexy, 2)
       ELSE
        znow = mapl(hexx, hexy, 3) + .9
      END IF
 'if the new elevation is less than the observer-target line
'goto the next terrain cell
      IF znow <= z1 + i * slope THEN GOTO 165

'if the terrain in the current cell blocks los, start drawing
'a line to indicate los is blocked
      xold = xn: yold = yn: t = i + 1
      slopenow = (znow - z1) / i
'loop checks to see if some terrain can be seen along the
'observer-target line, event though the line is blocked

        FOR j = t TO r
          xn = x + j * COS(k): yn = y + j * SIN(k)
          IF 0 >= xn OR xn >= 1000 THEN GOTO 166
          IF 0 >= yn OR yn >= 1000 THEN GOTO 166
          IF INT(xn / 20 + 1) = hexx AND INT(yn / 20 + 1) =
&           hexy THEN GOTO 164
          hexx = INT(xn / 20 + 1): hexy = INT(yn / 20 + 1)
'continue to check if the observer-target line is in the
'wooded area
          IF w = 1 AND mapl(hexx, hexy, 2) = 1 THEN w = 0
          IF w = 0 AND mapl(hexx, hexy, 2) < 1 THEN
 znext = (mapl(hexx, hexy, 3) + 10 / (mapl(hexx, hexy, 2)))
          ELSE
           znext = (mapl(hexx, hexy, 3) + .9)
          END IF
 'if the los is blocked draw a line
         IF znext < znow THEN GOTO 162

'if the elevation is the new cell is higher than the  previous
'terrain (i.e. the obsever can see it)go to the next terrain
'cell to see if the observer can see it
          IF slopenow > (znext - zl) / j THEN GOTO 162
          slopenow = (znext - zl) / j
          znow = znext
          xold = xn: yold = yn
          GOTO 164
 'if the terrain cannot be seen, draw a line
    162  LINE (xold, yold)-(xn, yn), 0
          xold = xn: yold = yn
   164 NEXT j
       GOTO 166
165 NEXT i
w = wl
166 NEXT k
INPUT "Do you want to continue? Y/N"; ans$
IF ans$ = "Y" OR ans$ = "y" THEN GOTO 160
END SUB


SUB map (opt%)
'****************************************************************
'MAP draws the screen map.  It also calls the map editor
'subprograms referenced by opt%
'****************************************************************

SHARED l 'l is the flag that defines whether an obstacle has
'been emplaced by the user

CLS
SCREEN 9, , 0, 0
WIDTH 80, 43
WINDOW (0, 0)-(1000, 1000)
'paint the screen the white background color
PAINT (500, 500), 15
' draw the wooded areas
```

```
LINE (40, 0)-(0, 380), 2, BF
LINE (40, 0)-(160, 360), 2, BF
LINE (160, 0)-(200, 320), 2, BF
LINE (200, 0)-(220, 300), 2, BF
LINE (220, 0)-(280, 280), 2, BF
LINE (280, 0)-(283, 280), 9, BF
'draw the stream
LINE (220, 280)-(283, 283), 9, BF
LINE (220, 280)-(223, 300), 9, BF
LINE (200, 300)-(220, 303), 9, BF
LINE (0, 380)-(40, 383), 9, BF
LINE (40, 360)-(160, 365), 9, BF
LINE (40, 383)-(43, 360), 9, BF
LINE (160, 360)-(163, 320), 9, BF
LINE (160, 320)-(200, 323), 9, BF
LINE (200, 323)-(203, 300), 9, BF

'draw the angled red roads
FOR i = 1 TO 25
LINE ((i - 1) * 20, 500 - 2 * (i - 1))-(i * 20, 500 - 3 - 2 *
(i - 1)), 4, BF
NEXT i
FOR i = 25 TO 50
LINE ((i - 1) * 20, 449 + 3 * (i - 25))-(i * 20, 449 - 3 + 3
* (i - 25)), 4, BF
NEXT i
LINE (500, 0)-(505, 1000), 0, B
FOR i = 0 TO 1000 STEP 40
LINE (501, i)-(504, i + 20), 4, BF
NEXT i
LINE (480, 455)-(480, 465), 0
LINE (160, 840)-(240, 880), 2, BF
FOR i = 1 TO 50
LINE (480 - (i - 1) * 15, 465 + (i - 1) * 20)-(480 - i * 15,
465 + i * 20), 0
NEXT i
LINE (330, 665)-(250, 665), 0
FOR i = 1 TO 25
LINE (250 - (i - 1) * 20, 665 + (i - 1) * 8)-(250 - i * 20,
665 + i * 8), 0
NEXT i
FOR i = 1 TO 25
LINE (750 + (i - 1) * 10, 0 + (i - 1) * 15)-(750 + i * 10, 0
+ i * 15), 0
NEXT i
FOR i = 1 TO 25
LINE (500 + (i - 1) * 18, 750 - (i - 1) * 9)-(500 + i * 18,
750 - i * 9), 0
NEXT i

'draw the contour lines
CIRCLE (150, 880), 50, 6, , , .45
CIRCLE (150, 900), 200, 6, , , .5
CIRCLE (150, 900), 600, 6, , , .2
CIRCLE (1000, 800), 1100, 6, , , .3
LINE (1000, 840)-(900, 720), 2, BF
CIRCLE (1000, 800), 70, 6, , , .2
CIRCLE (1000, 800), 120, 6, , , .25
CIRCLE (50, 50), 100, 6, , , .5
CIRCLE (50, 50), 200, 6, , , .3

' if an obstacle is present draw it, else skip this portion
IF l = 0 THEN GOTO 170
' loop to draw an obstacle
FOR i = 1 TO l
   LINE (lin(i, 1), lin(i, 2))-(lin(i, 3), lin(i, 4)), 0
   m! = (lin(i, 4) - lin(i, 2)) / (lin(i, 3) - lin(i, 1))
   b = lin(i, 2) - m * lin(i, 1)

'loop t draw the cross hatching on the wire
      FOR j = 0 TO (lin(i, 3) - lin(i, 1)) STEP 20
        x = lin(i, 1) + j
        y = lin(i, 2) + m * j
        LINE (x - 4, y + 4)-(x + 4, y - 4), 0
        LINE (x + 4, y + 4)-(x - 4, y - 4), 0
      NEXT j
NEXT i

'draw the magenta grid lines every 200 meters
170 FOR i = 200 TO 800 STEP 200
LINE (i, 0)-(i, 1000), 13
LINE (0, i)-(1000, i), 13
NEXT i

'copy the map to the hidden screen
PCOPY 0, 1

'call the approriate subprogram based on user input options

ON opt% GOTO 171, 172, 173, 174

171 LOCATE 1, 1: PRINT "HIT <CR> to continue."  'view map
ans$ = INPUT$(1)
GOTO 175

172 CALL wire                                   'emplace obstacle
 LOCATE 1, 1: PRINT "HIT <CR> to continue."
 ans$ = INPUT$(1)
GOTO 175

173 CALL contour                     'view contour interval data
 LOCATE 1, 1: PRINT "Hit <CR> to exit map"
 e$ = INPUT$(1)
 GOTO 175

174 CALL los                         'line of sight checks
```

```
175 END SUB

SUB mapp
'***********************************************************
'MApp is the main menu for the terrain editor.  It loads the
'default data into data arrays, allows the user to view data,
'input obstacles, and creates the .exp file for SPARTAN.
'***********************************************************

CLS
OPEN "mapl.dat" FOR INPUT AS #1
COLOR 15, 1
CLS
LOCATE 10, 20: PRINT "STANDBY WHILE TERRAIN ARRAY IS LOADED"

'load data array with default data
FOR i = 1 TO 50
   FOR j = 1 TO 50
      INPUT #1, mapl(i, j, 1), mapl(i, j, 2), mapl(i, j, 3)
   NEXT j
NEXT i
CLOSE #1

DO
CLS
left% = 10: right% = 70: top% = 4: bottom% = 22: fore% = 15:
&    back% = 1
WIDTH 80, 25
CALL frame(left%, right%, top%, bottom%, fore%, back%)
LOCATE 6, 20: PRINT "TERRAIN DATA FILE EDITOR MENU"
LOCATE 8, 20: PRINT "1) Look at map"
LOCATE 10, 20: PRINT "2) Add wire obstacle"
LOCATE 12, 20: PRINT "3) View elevation data and contour
&    levels"
LOCATE 14, 20: PRINT "4) View terrain data"
LOCATE 16, 20: PRINT "5) Line of sight"
LOCATE 18, 20: PRINT "6) Terrain Editor Help"
LOCATE 20, 20: PRINT "7) Exit to main menu"
6 LOCATE 21, 20: ch$ = INPUT$(1)
SELECT CASE ch$
CASE "1"
  COLOR 15, 0
  CALL map(1)
CASE "2"
  COLOR 15, 0
  CALL map(2)
CASE "3"
  COLOR 15, 0
  CALL map(3)
CASE "4"
  CALL TERRAINDAT
CASE "5"
```

```
    COLOR 15, 0
    CALL map(4)
  CASE "6"
  CALL maphelp
  CASE "7"
  CLS
  CALL frame(10, 70, 4, 7, 15, 1)
  LOCATE 5, 25: PRINT "STANDBY WHILE DATA FILE LOADS"

  'load edited data into .exp file

  OPEN "mapl.exp" FOR OUTPUT AS #2
  FOR i = 1 TO 50
     FOR j = 1 TO 50
       WRITE #2, mapl(i, j, 1), mapl(i, j, 2), mapl(i, j, 3)
     NEXT j
  NEXT i
  CLOSE #2
  EXIT DO
  CASE ELSE
    BEEP
     LOCATE 20, 20: PRINT "Try again, GOMER"
     GOTO 6
  END SELECT
  LOOP
  END SUB



SUB opening
'**********************************************************
' Opening screen for the preprocessor
'**********************************************************

CLS
left% = 1: right% = 80: top% = 3: bottom% = 22: fore% = 15:
&    back% = 9
CALL frame(left%, right%, top%, bottom%, fore%, back%)
left% = 9: right% = 72: top% = 10: bottom% = 16: fore% = 15:
&    back% = 9
CALL frame(left%, right%, top%, bottom%, fore%, back%)
LOCATE 13, 25: PRINT "     SPARTAN II COMBAT MODEL"
LOCATE 19, 22: INPUT "Press <Enter> when ready to continue",
&   start
END SUB

SUB phit
'*************************************************************
'PHIT is the subprogram that is the Phit editor main menu.  It
'reads the default data files into arrays, calls to various
'editor functions as the user inputs choices, and creates the
'.exp data files for SPARTAN
'*************************************************************
```

```
CLS
COLOR 15, 9
CLS
LOCATE 10, 20: PRINT "STANDBY WHILE ACCURACY DATA IS LOADED"
OPEN "m16.dat" FOR INPUT AS #1
OPEN "ak74.dat" FOR INPUT AS #2
OPEN "saw.dat" FOR INPUT AS #3
FOR i = 1 TO 8
  FOR j = 1 TO 4
    INPUT #1, p1(i, j)
    INPUT #2, p2(i, j)
    INPUT #3, p3(i, j)
  NEXT j
NEXT i
 CLOSE #1: CLOSE #2: CLOSE #3
DO
CLS
left% = 10: right% = 70: top% = 4: bottom% = 18: fore% = 15:
&    back% = 9
WIDTH 80, 25
CALL frame(left%, right%, top%, bottom%, fore%, back%)
LOCATE 6, 20: PRINT "WEAPON ACCURACY DATA FILE EDITOR MENU"
LOCATE 8, 20: PRINT "1) Review accuracy data"
LOCATE 10, 20: PRINT "2) Add weapon type"
LOCATE 12, 20: PRINT "3) Phit Editor HELP"
LOCATE 14, 20: PRINT "4) Exit to main menu"
3 LOCATE 21, 20: ch$ = INPUT$(1)
SELECT CASE ch$
CASE "1"
  CALL cphit
CASE "2"
  CALL addwpn
CASE "3"
CALL phithelp
CASE "4"
CLS
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 25: PRINT "STANDBY WHILE DATA FILE LOADS"
OPEN "P1HIT.exp" FOR OUTPUT AS #4
OPEN "P2HIT.exp" FOR OUTPUT AS #5
OPEN "P3HIT.exp" FOR OUTPUT AS #6
FOR i = 1 TO 8
FOR j = 2 TO 4
 PRINT #4, p1(i, j)
 PRINT #5, p2(i, j)
 PRINT #6, p3(i, j)
 NEXT j
 NEXT i
CLOSE #4
CLOSE #5
CLOSE #6
EXIT DO
CASE ELSE
```

```
  BEEP
   LOCATE 16, 20: PRINT "Try again, GOMER"
GOTO 3
END SELECT
LOOP
END SUB


SUB red (opt%, r)
'***********************************************************
'RED serves the same function for the RED soldiers that BLUE
'does for BLUE forces.  It prints out read soldier attribute
'values and allows the user to view, add, delete or edit those
'values.
'***********************************************************

CLS
WIDTH 80, 25
CALL frame(10, 70, 4, 7, 15, 9)

'view current soldier attribute values

LOCATE 5, 30: PRINT "RED SOLDIER DATA"
LOCATE 8, 1: PRINT "SOLDIER     DUTY      LOCATION   MOVEMENT
    STATUS    POSTURE    WEAPON"
LOCATE 9, 1: PRINT "         POSITION             DIRECTION
  "
j = 0
FOR i = 1 TO 12
 IF soldat(i, 1) > -1 THEN GOTO 216
 j = j + 1
 LOCATE 10 + j, 3: PRINT i
 LOCATE 10 + j, 11: ON soldat(i, 2) GOTO 201, 202, 203, 204,
205
201 PRINT "SQD LDR"
    GOTO 206
202 PRINT "TEAM LDR"
    GOTO 206
203 PRINT "GRENADIER"
    GOTO 206
204 PRINT "SAW GUNNER"
    GOTO 206
205 PRINT "RIFLEMAN"

206 LOCATE 10 + j, 23: PRINT USING "###"; soldat(i, 3);
soldat(i, 4)
LOCATE 10 + j, 36: PRINT USING "###"; soldat(i, 8) * 180 /
3.1415
LOCATE 10 + j, 46:
 IF soldat(i, 9) = 0 THEN
 PRINT "STATIONARY"
  GOTO 207
END IF
```

```
PRINT "MOVING"
207 LOCATE 10 + j, 57: ON soldat(i, 10) GOTO 208, 209, 210
208 PRINT "STANDING"
GOTO 211
209 PRINT "CROUCHING"
GOTO 211
210 PRINT "PRONE"
211 LOCATE 10 + j, 69: ON soldat(i, 11) GOTO 212, 213, 214,
215
212 IF soldat(i, 2) = 3 THEN
PRINT "M203"
GOTO 216
END IF
PRINT "M16A2"
GOTO 216
213 PRINT "AK-74"
GOTO 216
214 PRINT "SAW"
GOTO 216
215 PRINT "OTHER"
216 NEXT i
LOCATE 1, 25: PRINT "HIT <CR> to continue"
e$ = INPUT$(1)

'based on user choice: opt%=0---depart editor
                            1---delete one or more red soldiers
                            2---add one or moore soldiers

IF opt% = 0 THEN GOTO 221
IF opt% = 2 THEN GOTO 218

'delete a soldier

217 LOCATE 11 + j: INPUT "Which soldier do you want to
delete?", d%
IF soldat(d%, 1) > -1 THEN
PRINT "The number you input is not a red soldier, try again."
GOTO 217
END IF
FOR i = 1 TO 15
soldat(d%, i) = 0
NEXT i
GOTO 221

' Add one or more soldiers

218 CLS
LOCATE 2, 20: PRINT "INPUT RED SOLDIER DATA"
LOCATE 6, 4: PRINT "DUTY POSITION"
LOCATE 7, 4: PRINT "X GRID COORD"
LOCATE 8, 4: PRINT "Y GRID COORD"
LOCATE 9, 4: PRINT "#GRENADES"
LOCATE 10, 4: PRINT "TIME FIRED"
LOCATE 11, 4: PRINT "NOT USED"
LOCATE 12, 4: PRINT "MOVEMENT DIRECTION"
LOCATE 13, 4: PRINT "MOVEMENT STATUS"
LOCATE 14, 4: PRINT "POSTURE"
LOCATE 15, 4: PRINT "WEAPON TYPE"
LOCATE 16, 4: PRINT "ROUNDS PER MAGAZINE"
LOCATE 17, 4: PRINT "NUMBER MAGAZINES"
LOCATE 18, 4: PRINT "TARGET ID"
LOCATE 19, 4: PRINT "WOUND STATUS"
FOR i = 1 TO r
FOR j = 1 TO 12
IF soldat(j, 2) > 0 THEN GOTO 219
LOCATE 3, 18 + i * 10: PRINT "SOLDIER"; i
soldat(j, 1) = 0
FOR k = 2 TO 15
 LOCATE 3 + k, 18 + i * 10: INPUT soldat(j, k)
NEXT k
GOTO 220
219 NEXT j
220 NEXT i
221 END SUB


SUB SOLDIER
'**********************************************************
'SOLDIER is the main menu for the soldier attribute editor
'It reads default data into data arrays, allows the user to
'access editor functions, and reads the altered data array
'into the .exp file for SPARTAN.
'**********************************************************

CLS
OPEN "joe.dat" FOR INPUT AS #1
COLOR 15, 9
CLS
LOCATE 10, 20: PRINT "STANDBY WHILE SOLDIER DATA ARRAY IS
LOADED"

FOR i = 1 TO 12    'read data file
  FOR j = 1 TO 15
    INPUT #1, soldat(i, j)
  NEXT j
NEXT i
CLOSE #1

DO
CLS
left% = 10: right% = 70: top% = 4: bottom% = 23: fore% = 15:
back% = 9
WIDTH 80, 25
CALL frame(left%, right%, top%, bottom%, fore%, back%)
LOCATE 6, 20: PRINT "SOLDIER DATA FILE EDITOR MENU"
LOCATE 8, 20: PRINT "1) Look at Blue Squad"
LOCATE 10, 20: PRINT "2) Look at Red Squad"
LOCATE 12, 20: PRINT "3) Delete soldiers"
LOCATE 14, 20: PRINT "4) Add soldiers"
LOCATE 16, 20: PRINT "5) Edit soldier attributes"
LOCATE 18, 20: PRINT "6) Pick formation and location"
LOCATE 20, 20: PRINT "7) Soldier Editor Help"
LOCATE 22, 20: PRINT "8) Exit to main menu"
7 LOCATE 21, 20: ch$ = INPUT$(1)
SELECT CASE ch$
CASE "1"
  CALL blue(0, 0)
CASE "2"
  CALL red(0, 0)
CASE "3"
  CALL delete
CASE "4"
  CALL add
CASE "5"
  CALL joeatrib
CASE "6"
  CALL format
CASE "7"
CALL joehelp
CASE "8"

 OPEN "joe.exp" FOR OUTPUT AS #2   'create .exp file
 FOR i = 1 TO 12
   FOR j = 1 TO 15
     PRINT #2, soldat(i, j)
   NEXT j
 NEXT i
 CLOSE #2

EXIT DO
CASE ELSE
  BEEP
   LOCATE 22, 20: PRINT "Try again, GOMER"
GOTO 7
END SELECT
LOOP
END SUB


SUB TERRAINDAT
'*************************************************************
'TERRAINDAT allows the user to view the values of the terrain
'cell's three attributes
'*************************************************************

CLS
CALL frame(10, 70, 4, 7, 15, 9)
```

```
LOCATE 5, 30: PRINT "TERRAIN DATA FILE"
LOCATE 11, 1: PRINT "X HEX         Y HEX"
LOCATE 12, 1: PRINT "INDEX         INDEX"
LOCATE 11, 26: PRINT " MOBILITY     ATTENUATION    ELEVATION"
LOCATE 12, 26: PRINT "  FACTOR        FACTOR        (meters)"
x = 0

'this controls the amount of cell data appearing on the screen
'to prevent scrolling

VIEW PRINT 13 TO 24
FOR i = 1 TO 50  'load terrain data matrix
   FOR j = 1 TO 50
      x = x + 1
      PRINT i, j, mapl(i, j, 1), mapl(i, j, 2), mapl(i, j, 3)
      IF x = 10 THEN GOTO 231
   230 NEXT j
NEXT i

GOTO 232
231 PRINT "Do you want to continue? Y/N"
ans$ = INPUT$(1)
IF ans$ = "y" THEN
x = 0
GOTO 230
END IF
232 VIEW PRINT
END SUB


SUB wire
'************************************************************
'WIRE allows the user to input a wire obstacle and to view it.
'It also creates the obstacle file and alters the mobility
'factor of the terrain cells through which the obstacle
'passes.
'************************************************************

SHARED l
PCOPY 1, 0
VIEW (140, 150)-(550, 200), 9, 0
l = 0
240 l = l + 1

'query the user for input

LOCATE 20, 20, 0: PRINT "Input obstacle starting X and Y
coordinates"
LOCATE 21, 22, 0: PRINT "For example <725,856> ."
LOCATE 21, 45, 0: INPUT xl, yl
LOCATE 22, 20, 0: PRINT "Input obstacle ending X and Y
&   coordinates"
```

```
LOCATE 23, 22, 0: PRINT "For example  <345,999> ."
LOCATE 23, 45, 0: INPUT x2, y2
LOCATE 25, 20, 0: PRINT "To view obstacle hit <CR>"
LOCATE 26, 20, 0: e$ = INPUT$(1)
VIEW
PCOPY 1, 0

'ensure the obstacle is not vertical (infinite slope)

IF x1 = x2 THEN x2 = x2 + 1
IF x1 > x2 THEN
   x = x1: x1 = x2: x2 = x
   y = y1: y1 = y2: y2 = y
END IF

 lin(1, 1) = x1: lin(1, 2) = y1: lin(1, 3) = x2: lin(1, 4)=y2

'draw the obstacle
LINE (x1, y1)-(x2, y2), 0

'compute equation of obstacle
m = (y2 - y1) / (x2 - x1)
b = y1 - m * x1

'loop draws cross hatching on obstacle
FOR i = 0 TO ABS(x2 - x1) STEP 20
   x = x1 + i
   y = y1 + m * i
   LINE (x - 4, y + 4)-(x + 4, y - 4), 0
   LINE (x + 4, y + 4)-(x - 4, y - 4), 0
NEXT i

PCOPY 0, 1

'loop to update mobility factor of terrain cells
FOR k = x1 TO x2
  x = INT(k / 20 + 1)
  y = INT((m * k + b) / 20 + 1)
  map1(x, y, 1) = .1
NEXT k

PCOPY 0, 1
LOCATE 1, 1: PRINT "Hit <CR> to return to Terrain Editor Menu"
e$ = INPUT$(1)
PCOPY 1, 0

OPEN "obs.exp" FOR OUTPUT AS #1   'create obstacle file
FOR i = 1 TO 1
WRITE #1, lin(i, 1), lin(i, 2), lin(i, 3), lin(i, 4)
NEXT i
CLOSE #1
END SUB
```

Appendix F: Simulation Code

This appendix contains the program code for the SPARTAN simulation model. The code is written in QuickBASIC 4.5.

The code is contained in three separate modules. SPARTAN.bas module contains the simulation code. The module display.bas contains all the subprograms that deal with user requested status updates. The module mainhelp.bas contains all the help files. DISPLAY and MAINHELP are not included in this appendix because they consist solely of formatted display screens of text and data arrays. In addition, all information contained in the help screens is in the thesis in Chapter IV and in the user's manual.

The same notation used in Appendix E is in effect in this appendix.

```
'*************************************************************
'*                                                           *
'*                SPARTAN Simulation Code                    *
'*                   **Main Module**                         *
'*                                                           *
'*************************************************************

'*************************************************************
'The main module contains all subprograms that make the
'simulation work: the outer execution loop, initialization
'programs, and the subprograms for all combat processes.
The 'primary function of this first subprogram is to define
all 'subprograms, arrays, and to define the terminating
'conditions.  This module also contains the logic that
'terminates the simulation.
'*************************************************************

'This section declares all subprograms and functions in used
'in the simulation.

DECLARE SUB soldier ()  'displays current soldier attribute
                        'values
DECLARE SUB schevent () 'displays all events on calendar
DECLARE SUB pottgt () 'displays current detected target list
DECLARE SUB battlestat ()  'displays battle statistics
DECLARE SUB killcard () 'displays hit information per weapon
                        'type
DECLARE SUB explain ()  'displays information about how to
                        'use SPARTAN
DECLARE SUB maindisplay () 'main menu for all user requested
                           'reports
DECLARE SUB explode (x!, y!, r!) 'draws explosions on screen
DECLARE FUNCTION triag! (a!, d!, B!) 'triangle function
DECLARE SUB evnt () 'pulls next scheduled event off calendar
DECLARE SUB move (ind!, tnow)       'moves soldiers
DECLARE SUB frame (left%,right%,top%,bottom%, fore%, back%)
DECLARE SUB init ()         'initializes all data arrays
DECLARE SUB map ()          'draws map
DECLARE SUB schedule (act, ind, T)  'adds events to calendar
DECLARE SUB los (obs, tgt, x, y, x2, y2, r)  'checks line of
                                     ' sight from observer to target
DECLARE SUB acquire (obs, time) 'target detection subprogram
DECLARE SUB selct (obs, time)   'target selection subprogram
DECLARE SUB shoot (obs, time)   'direct fire engagement
                                'subprogram
DECLARE SUB wire (obs)        'obstacle detection subprogram
DECLARE SUB breach (side, tnow)'initiates obstacle breaching
DECLARE SUB endbreach (ind, time)'moves squad through breach
DECLARE SUB refresh ()            'refreshes screen
DECLARE SUB impact (tgt) 'determines results of round impact
DECLARE SUB react (tgt, time)   'determines reaction to fire
DECLARE SUB direction (obs, time) 'changes squad's direction
DECLARE SUB indirect (obs, time)  'indirect fire engagements
DECLARE SUB adjust ()         'alters terminating conditions

'This section defines common arrays and variables that are
'used across all modules.

' soldat contains all soldier attribute values
' event is the current event calendar
' ptgt is the matrix of potential targets
' bluecount is the starting number of blue soldiers
' redcount is the starting number of red soldiers
' activeblue is the number of blue soldiers remaining
' activered is the number of red soldiers remaining
' timetostop is the flag to stop the simulation '

COMMON SHARED soldat(), event(), ptgt(), tgtrec(),
bluecount, &  redcount, activeblue, activered, timetostop

'obs = observer ID
'time = current simulation time
'rwire = flag of RED identifying presence of obstacle
'bwire = flag of BLUE identifying presence of obstacle
'rbrch = flag of RED breach status
'bbrch = flag of BLUE breach status
COMMON SHARED obs, time, rwire, bwire, bbrch, rbrch

'DYNAMIC creates the data arrays outside the 64K set aside
by DOS for program execution
'$DYNAMIC

'This section defines the data arrays
DIM SHARED soldat(12, 17) 'matrix of soldier attributes
DIM SHARED event(99, 3)   'event calendar
DIM SHARED ptgt(12, 12)   'potential target list
DIM SHARED tgtrec(8, 4)   'weapons' hit data record
DIM SHARED barray(0 TO 102) 'array containing blue icon
DIM SHARED rarray(0 TO 102) 'array containing red icon
DIM SHARED array2(0 TO 102) 'array icon to erase old
 '                             soldier's positions
DIM SHARED darray(0 TO 102) 'array containing dead icon
DIM SHARED lin(10, 4)       'obstacle matrix
DIM SHARED thresh(12, 12)  'random threshold observer-target
 '                              Pinf values
DIM SHARED woods(10, 4)   'matrix of Pinf values for targets
 '                              in wooded areas
DIM SHARED nowoods(10, 4) 'matrix of Pinf values for targets
 '                              not in wooded areas
DIM SHARED corl(10, 4)    'cycles resolvable by the observer
 '                           for targets not in wooded areas
DIM SHARED p1(8, 3)        'M16A2 Phit table
DIM SHARED p2(8, 3)        'AK74 Phit table
DIM SHARED p3(8, 3)        'SAW/RPK74 Phit table
DIM SHARED p5(8, 3)        'User input Phit table
DIM SHARED p6(8, 3)        'User input Phit table
DIM SHARED mapl(50, 50, 3)   'terrain cell data

'This section identifies default terminating conditions
'termevnt = number of events processed
'timestop = simulation time passed
'bluestop = blue soldiers remaining
'redstop = red soldiers remaining

DIM SHARED termevnt, timestop, bluestop, redstop
LET termevnt = 5000: LET timestop = 350: LET bluestop = 6:
LET &    redstop = 1

'turn on function keys
'  F(1) accesses the main display menu
ON KEY(1) GOSUB 1000
KEY(1) ON
'  F(2) refreshes the screen
ON KEY(2) GOSUB 2000
KEY(2) ON

'Open the history file
OPEN "history.dat" FOR OUTPUT AS #10
COLOR 15, 9

'Initialize the data arrays
CALL init

'Display expanatory screen
CALL explain
SCREEN 9, , 1, 1
WIDTH 80, 43

'draw the map
CALL map

'initialize startime which is used to tie simulation clock
'advancement to passage of real time
starttime = TIMER
timeon = 1

quit = 0: timetostop = 0: activeblue = bluecount: activered
= &  redcount

'Continue processing events until one of the termirating
'conditions is met

DO WHILE timetostop = 0
    CALL evnt          'pull the next event off the calendar
    quit = quit + 1      'count number of events processed

'check terminating conditions
    IF (quit >= termevnt OR time >= timestop OR activeblue
&    <=bluestop OR activered <= redstop) THEN timetostop = 1
```

```
LOOP

'close history file
CLOSE #10
CLS
LOCATE 10, 4: INPUT "Do you want to see final results?Y/N",
&   ans$
IF ans$ = "Y" OR ans$ = "y" THEN

'show final results if user requests
CALL soldier
CALL schevent
CALL pottgt
CALL battlestat
CALL killcard
END IF
9  END
1000 :'Turn off advancement of real time while display is
      'active.  Resume time advancement when control returns
'      to the simulation
displaytime = TIMER
CALL maindisplay
 CLS
 SCREEN 9
 COLOR 15, 0
 CALL map
starttime = starttime + (TIMER - displaytime)
RETURN


2000 :
CALL refresh
RETURN


SUB acquire (obs, time)
'*************************************************************
'ACQUIRE is the subprogram that checks all three conditions
'of target acquisition.  If the observer is a squad leader
'and there is an obstacle present, the routine calls to WIRE
'to see if the obstacle is detected.  The routine then
'checks all enemy soldiers to see if detection is possible.
'If, at any step, detection fails, then the routine begins
'checking the next enemy soldier until all enemy soldiers
'have been checked.
'*************************************************************
'VARIABLES:
' nexttime = variable to define start search time for event
' losl = flag to denote line of sight
' bwire = flag for BLUE obstacle detection
' rwire = flag for RED obstacle detection
'sel = flag to denote whether any tarets were selected
```

```
'  obs = observer ID
'  tgt= target ID

SHARED nexttime, losl, bwire, rwire
sel = 0

'check soldier for nonmove status or to see if he is dead

IF soldat(obs, 15) = 0 OR soldat(obs, 1) = 0 THEN
    nexttime = time + 100
    GOTO 98
END IF

'if soldier is not a squad leader continue
IF soldat(obs, 2) > 1 THEN GOTO 95

'else check for obstacle
 IF soldat(obs, 1) > 0 AND bwire < 2 THEN GOTO 95
 IF soldat(obs, 1) < 1 AND rwire < 2 THEN GOTO 95
  CALL wire(obs)


'loop for target detection
95 FOR i = 1 TO 12

'if target is the same side or is not a soldier then check
'next soldier

  IF soldat(obs, 1) = soldat(i, 1) OR soldat(i, 1) = 0 THEN
99
 'assign observer location
  x1 = soldat(obs, 3): y1 = soldat(obs, 4)
'assign target location
  x2 = soldat(i, 3): y2 = soldat(i, 4)
'determine observer-target range
  range = (((x1 - x2) ^ 2 + (y1 - y2) ^ 2) ^ .5) / 1000

'if range greater than 1000 meters check next target
  IF range > 1 THEN GOTO 99

'if range less than 50 meters then change both target and
'observer to nonmoving status
  IF range * 1000 < 50 THEN
     soldat(obs, 9) = 0
     soldat(i, 9) = 0
  END IF

'if the observer has already acquired this target go and
check 'line of sight

 IF (ptgt(obs, i) > 0) THEN GOTO 91

 'assign critical value = threshold value for that observer-
 ' target pair
 crit = thresh(obs, i)


'determine if the target is in a wooded area
  hexx = INT(soldat(i, 3) / 20 + 1)
  hexy = INT(soldat(i, 4) / 20 + 1)
  IF (mapl(hexx, hexy, 1) < 1) THEN GOTO 93

'adjust target dimensions according to target move status
and
' posture

'target non moving
   IF soldat(i, 9) < 1 THEN
     IF soldat(i, 10) = 3 THEN      'tgt is prone
       pinf = nowoods(INT(range * 10 + .5), 2)
       cor = corl(INT(range * 10 + .5), 2)
       GOTO 94
       ELSE        'tgt is crouched or standing
       pinf = nowoods(INT(range * 10 + .5),1)
       cor = corl(INT(range + 10 + .5),1)
       GOTO 94
     END IF
    END IF

'else the soldier is moving
     IF soldat(i,10) = 3 THEN
       pinf = nowoods(INT(range * 10 + .5), 4)
       cor = corl(INT(range * 10 + .5), 4)
       GOTO 94
     ELSE
       pinf = nowoods(INT(range * 10 + .5),3)
       cor = corl(INT(range + 10 + .5),3)
       GOTO 94
     END IF


'target is in the woods
93 IF soldat(i, 9) < 1 THEN
     IF soldat(i, 10) = 3 THEN
      pinf = woods(INT(range * 10 + .5), 2)
      cor = corl(INT(range * 10 + .5), 2) * .775
      GOTO 94
     ELSE
      pinf = woods(INT(range * 10 + .5), 1)
      cor = corl(INT(range * 10 + .5), 1) * .775
      GOTO 94
     END IF
   END IF
     IF soldat(i, 10) = 3 THEN
      pinf = woods(INT(range * 10 + .5), 4)
      cor = corl(INT(range * 10 + .5), 4) * .775
      GOTO 94
     ELSE
      pinf = woods(INT(range * 10 + .5), 3)
      cor = corl(INT(range * 10 + .5), 3) * .775
      GOTO 94
   END IF

'if the Pinf is less than the threshold, then no detetection
is possible, go to the next target

94 IF pinf < thresh(obs, i) THEN
   ptgt(obs, i) = 0
   GOTO 99
   END IF


'If detection is possible, check line of sight
'obs = observer ID, i = target id

91 CALL los(obs, i, x1, y1, x2, y2, range)
 'if los does not exist, the ptgt value is zeroed out
 ' and the observer checks the next target

    IF (losl < 1) THEN
       ptgt(obs, i) = 0
       GOTO 99
    END IF

'If detection is possible, and line of sight exists, then
'see if the observer can detect the target

'if the observer has already detected the target, check the
'next target
IF ptgt(obs, i) > 0 THEN GOTO 99

'if the target is moving or has fired the last 20 time
'units, adjust the target dimensions

IF soldat(i, 9) > 0 OR (time - soldat(i, 6)) < 20 THEN cor =
&   cor * 2

'compute pfov

pfov = 1 - EXP(-1 / 6.8 * cor * (1.7 + 3.4 * (RND)))

'if the observer has already once to detect the target or if
'a squad member has already detected the target adjust the
'Pdet

pdet = pinf * pfov + ABS(ptgt(obs, i))

'Bernoulli trial to see if detection is made
IF (RND > pdet) THEN
```

```
 ' no detection, annotate the potential target list
 'check the next target
 ptgt(obs, i) = -.5 * pdet
  GOTO 99
END IF

'detection, annotate the potential target list

ptgt(obs, i) = pdet

'loop to adjust the pdet values for squadmates
 FOR k = 1 TO 12
    'don't annotate the other side
  IF k = i OR soldat(k, 1) <> soldat(obs, 1) THEN GOTO 92
  IF ptgt(k, i) > -.5 AND ptgt(k, i) <= 0 THEN
  ptgt(k, i) = ptgt(k, i) - .4 * pdet
  END IF
92 NEXT k

'display message on the screen reflecting target detection
IF (soldat(obs, 1) > 0) THEN
 strng$ = "Blue"
 ELSE
 strng$ = "Red "
 END IF
 LOCATE 1, 1
 PRINT USING "&  soldier  detects  enemy at ### ###";
&   strng$; soldat(i, 3); soldat(i, 4)

'adjust sel so after all enemy soldiers have been detected,
'a SELECT target event will be scheduled
sel = 1

99 NEXT i

'schedule a select target

IF sel = 1 THEN
 CALL schedule(2, obs, time + 5)
 GOTO 98
 END IF
 CALL schedule(1, obs, time + RND * 40)
98 END SUB
```

```
SUB adjust
'*************************************************************
'adjust allows the user to alter the simulation terminating
'conditions.  Users can alter the number of surviving
'soldiers, the number of events processed, or the simulation
'time.  Users can also turn off the linking of simulation
'time to real time and can dictate another random number
'seed.
'*************************************************************

SHARED bluecount, redcount, timeon
DO
CLS
WIDTH 80, 25
SCREEN 9
COLOR 15, 1
LOCATE 4, 25: PRINT "ADJUSTING TERMINATING CONDITIONS"
LOCATE 6, 4: PRINT "Initial terminating conditions are:"
LOCATE 8, 10: PRINT "1) Total number of events---5000"
LOCATE 9, 10: PRINT "2) Time to stop---350 time units"
LOCATE 10, 10: PRINT "3) 6 Blue soldiers remaining"
LOCATE 11, 10: PRINT "4) 1 Red soldier remaining"
LOCATE 12, 10: PRINT "5) Random number seed---0"
LOCATE 13, 10: PRINT "6) Timer on"
LOCATE 15, 4: INPUT "Input the condition you wish to change
or (7) to quit"; ans$
SELECT CASE ans$
CASE "1"
LOCATE 16, 4: INPUT "Input new number of events (> 50)"; n
IF n < 50 THEN
PRINT "Your input must be greater than 50, try again"
ELSE
termevnt = n
END IF
CASE "2"
LOCATE 16, 4: INPUT "Input new time to stop (> 50)"; n
IF n < 50 THEN
PRINT "Your input must be greater than 50, try again"
ELSE
timestop = n
END IF
CASE "3"
LOCATE 16,4:INPUT "Input new Blue soldiers remaining (> 0)"
n
n = INT(n)
IF n < 1 THEN
PRINT "Your input must be greater than 0, try again"
ELSE
bluestop = n
END IF
CASE "4"
LOCATE 16,4: INPUT "Input new Red soldiers remaining (> 0)"
n
```

```
n = INT(n)
IF n < 1 THEN
PRINT "Your input must be greater than 0, try again"
ELSE
redstop = n
END IF
CASE "5"
LOCATE 16, 4: INPUT "Input new random number seed (-32768 <
&     seed <32767"; seed
IF seed < -32768 OR seed > 32767 THEN
PRINT "Your seed value is out of range, try again"
ELSE
RANDOMIZE seed
END IF
CASE "6"
LOCATE 16, 4: INPUT "Do you want the simulation clock tied
to real time? Y/N"; a$
IF a$ = "Y" OR a$ = "y" THEN
timeon = 1
ELSE
timeon = 0
END IF
CASE ELSE
  EXIT DO
END SELECT
LOOP
END SUB


SUB breach (side, tnow)
'************************************************************
'BREACH adjusts the attributes of all squad members that are
'currently enmeshed in a breaching operation.  The
'attributes are changed to prone and nonmoving.  In
'addition, all moves for that side are deleted from the
'event calendar and an ENDBREACH scheduled.
'************************************************************

SHARED rbrch, bbrch

'display message

LOCATE 2, 1: PRINT "Breaching Obstacle"

'alter breaching flag to reflect breach in progress

IF side > 0 THEN bbrch = 2
IF side < 0 THEN rbrch = 2

'loop to delete all move events for that side

FOR i = 1 TO 99
```

```
  IF event(i, 1) <> 4 THEN GOTO 139
  IF soldat(event(i, 2), 1) = side THEN
    event(i, 1) = 0: event(i, 2) = 0: event(i, 3) = 0
  END IF
139 NEXT i

'loop to alter soldier attributes
FOR i = 1 TO 12
  IF soldat(i, 1) = side THEN
    soldat(i, 9) = 0
    soldat(i, 10) = 3:soldat(i, 7) = 3
  END IF
NEXT i

'schedule an ENDBREACH and a REFRESH screen
FOR i = 1 TO 12
   IF soldat(i, 1) = side THEN
      x = tnow + 100
      CALL schedule(5, i, x)
      CALL schedule(8, i, x - 50)
      GOTO 138
   END IF
NEXT i
138 END SUB



SUB direction (obs, time)
'*************************************************************
'DIRECTION is scheduled when a squad leader detects an
'enemy.  The subroutine checks if the squad leader's target
'is more than 25 degrees off the current azimuth.  If so, a
'direction change is directed for the squad.  Regardless of
'the direction, the squad is directed to come on line,
'oriented on the squad leader's direction cf travel.
'*************************************************************

IF soldat(obs, 1) = 0 THEN 202
SHARED nexttime, bbrch, rbrch, bwire, rwire

'identify the side making the breach

side = soldat(obs, 1)

'display message reflecting formation change

IF side > 0 THEN
 strng$ = "Blue"
 ELSE
 strng$ = "Red"
 END IF
 LOCATE 1, 1: PRINT USING "&    squad,    adjust   formation
```

```
  "; strng$

'compute observer-target azimuth

 x = soldat(obs, 3): y = soldat(obs, 4)
tgt = soldat(obs, 14)
x2 = soldat(tgt, 3): y2 = soldat(tgt, 4)
k = ATN(ABS((y2 - y) / (x2 - x)))
IF (y2 > y) AND (x2 < x) THEN k = 3.141 - k
IF (y2 < y) AND (x2 < x) THEN k = k + 3.141
IF (y2 < y) AND (x2 > x) THEN k = -k
IF (y1 = y) AND (x2 < x) THEN k = 3.141
dir = k

'check to see if azimuth is greater than 25 degrees

IF ABS(dir - soldat(obs, 8)) < .436 THEN
  'no direction change
   dir = soldat(obs, 8)
ELSE
  ' a direction change, if there is wire present, alter the
  ' breach flag so a new breach must be effected

  IF side > 0 AND bwire > 0 THEN
     bwire = 2
     IF bbrch < 2 THEN bbrch = 0
  END IF
  IF side < 0 AND rwire > 0 THEN
       rwire = 2
       IF rbrch < 2 THEN rbrch = 0
  END IF
END IF

'loop to update soldier location for new formation

 FOR i = 1 TO 12
  j = i
IF soldat(j, 1) = side AND soldat(j, 15) > 0 THEN
     soldat(i, 8) = dir
     IF i > 9 THEN j = i - 9
     IF j = 1 THEN
    soldat(i, 3) = x: soldat(i, 4) = y
ELSEIF j = 5 THEN
soldat(i, 3) = x + 36 * COS(2.55 + dir): soldat(i, 4) = y +
36 * SIN(2.55 + dir)
ELSEIF j = 3 THEN
soldat(i, 3) = x + 30 * COS(1.57 + dir): soldat(i, 4) = y +
30 * SIN(1.57 + dir)
ELSEIF j = 2 THEN
soldat(i, 3) = x + 56 * COS(1.39 + dir): soldat(i, 4) = y +
56 * SIN(1.39 + dir)
ELSEIF j = 4 THEN
soldat(i, 3) = x + 65.7 * COS(1.72 + dir): soldat(i, 4) = y
```

```
+ 65.7 * SIN(1.72 + dir)
ELSEIF j = 9 THEN
soldat(i, 3) = x + 36 * COS(-2.55 + dir): soldat(i, 4) = y +
36 * SIN(-2.55 + dir)
ELSEIF j = 7 THEN
soldat(i, 3) = x + 30 * COS(-1.57 + dir): soldat(i, 4) = y +
30 * SIN(-1.57 + dir)
ELSEIF j = 6 THEN
soldat(i, 3) = x + 56 * COS(-1.39 + dir): soldat(i, 4) = y +
56 * SIN(-1.39 + dir)
ELSEIF j = 8 THEN
soldat(i, 3) = x + 65.7 * COS(-1.72 + dir): soldat(i, 4) = y
+ 65.7 * SIN(-1.72 + dir)
END IF
END IF
NEXT i
202 nexttime = time + 100
201 END SUB



SUB endbreach (ind, time)
'*************************************************************
'ENDBREACH is scheduled for 100 time units after a breaching
'operation begins.  The subroutine updates the breaching
'flag and the changes the attributes of the breaching
'element to moving and standing.  A move is also scheduled.
'*************************************************************

SHARED nexttime, rbrch, bbrch
side = soldat(ind, 1)

'loop to update soldier attributes

FOR i = 1 TO 12
  IF soldat(i, 1) = side AND soldat(i, 15) > 0 THEN
     soldat(i, 10) = 1 : soldat(i, 7) = 1
     soldat(i, 9) = 1
     CALL schedule(4, i, time + .5 * i)
  END IF
149 NEXT i

'update breaching flags

IF side = 1 THEN
 bbrch = 1
 ELSE
 rbrch = 1
 END IF
nexttime = time + .5
END SUB
```

```
SUB evnt
'**************************************************************
'EVNT checks the event calendar for the next scheduled
'event, pulls it from the calendar, checks to see if the
'event should be processed or what until more "real time"
'has advanced, and then calls the subprogram referenced by
'the event.
'**************************************************************

SHARED nexttime, time, starttime, timeon

'loop to check all calendar event for the first scheduled
'event

24 FOR i = 1 TO 99
        'if the event ID = 0, it is empty
     IF event(i, 1) < 1 THEN GOTO 10

 'if the event time is less than nextime, it is the ranking
  'event rofor execution

    IF event(i, 3) <= nexttime THEN
        opt = i
        nexttime = event(i, 3)
   END IF
10 NEXT i

ind = event(opt, 2)
evt = event(opt, 1)
time = nexttime
'zero out the event array row for the  pulled event
event(opt, 1) = 0: event(opt, 2) = 0: event(opt, 3) = 0

'if the soldier was deleted, get another event

IF soldat(ind, 1) = 0 THEN
    nexttime = time + 100
    GOTO 24
END IF

'if the simulation clock is tied to real time check time
'advancement

IF timeon = 1 THEN
     x = (starttime + time) - TIMER
     IF x <= 1 THEN GOTO 22
     SLEEP INT(x)
END IF

'write the event to the history file

22 WRITE #10, evt, ind, time
```

```
'display event message on screen

LOCATE 1, 60: PRINT USING "Time now is ###.###"; time
LOCATE 2, 60: PRINT "EVENT "; evt; " ACTOR "; ind
IF time < T THEN e$ = INPUT$(1)
T = time

'call the subprogram referenced by the event

ON evt GOTO 11, 12, 13, 14, 15, 16, 17, 18, 19
11 CALL acquire(ind, time)
GOTO 23
12 CALL selct(ind, time)
GOTO 23
13 CALL shoot(ind, time)
GOTO 23
14 CALL move(ind, time)
GOTO 23
15 CALL endbreach(ind, time)
GOTO 23
16  CALL react(ind, time)
GOTO 23
17 CALL direction(ind, time)
GOTO 23
18 CALL refresh
GOTO 23
19 CALL indirect(ind, time)
23 END SUB


SUB explode (x, y, r)
'************************************************************
' EXPLODE draws explosions on the screen
'************************************************************
'VARIABLES:
' x is the horizontal coordinate
' y is the vertical coordinate
' r is the burst radius

PLAY "t80"
SOUND 250, 2.5
FOR i = 1 TO 25
    PLAY "164 nC"
NEXT i
PLAY "MBOOL32EFGEFDC"
Radius = r
FOR c# = 0 TO Radius STEP .5
    CIRCLE (x, y), c#, 4
NEXT c#
FOR i = 1 TO 50
    PLAY "164 n0"
NEXT i
END SUB
```

```
SUB impact (tgt)
'*****************************************************
'IMPACT determines the results of a hit on a target.  This
'subprogram is scheduled by either the SHOOT or the INDIRECT
'subprograms.
'*****************************************************

SHARED bluecount, redcount, activeblue, activered
x = soldat(tgt, 3): y = soldat(tgt, 4)
side = soldat(tgt, 1)

'if the soldier is already dead, process the next event

IF soldat(tgt, 15) < 1 THEN GOTO 179

'if the soldier is already wounded, his chances of being
'killed increase to 50%

IF soldat(tgt, 15) = 1 AND RND > .5 THEN
     GOTO 172
   ELSE
     GOTO 173
END IF

'if the soldier is not wounded, his chances of being killed
'are 30%
IF RND > .7 THEN

'change soldier status to dead, prone, nonmoving
  172 soldat(tgt, 15) = 0
      soldat(tgt, 10) = 3
      soldat(tgt, 9) = 0

'adjust the active soldier count

     IF soldat(tgt, 1) < 0 THEN activered = activered - 1
     IF soldat(tgt, 1) > 0 THEN activeblue = activeblue - 1

'loop to remove all scheduled events from event calendar

     FOR i = 1 TO 99
        IF event(i, 2) = tgt THEN
          event(i, 1) = 0: event(i, 2) = 0: event(i, 3) = 0
        END IF
     NEXT i

'change icon color

     PUT (x, y), darray, PSET

'if the soldier is a squad leader, change of command
```

```
      IF soldat(tgt, 2) = 1 THEN
         FOR i = 1 TO 12
           IF soldat(i, 1) <> side THEN GOTO 171
           IF soldat(i, 2) = 2 THEN
             soldat(i, 2) = 1
             GOTO 179
           END IF
    171 NEXT i
      END IF
   GOTO 179
   END IF

   'if the soldier is only wounded
   'update status to prone, nonmoving, wounded
   173 soldat(tgt, 15) = 1
   soldat(tgt, 10) = 3
   soldat(tgt, 9) = 0
   nexttime = 5000
   179 END SUB


   SUB indirect (tgt, time)
   '*************************************************************
   'INDIRECT processes all mortar fire missions.  It is
   'scheduled during the SELECT target event  when the BLUE
   'squad leader detects two or more targets.
   '*************************************************************

   x = soldat(tgt, 3): y = soldat(tgt, 4)

   'assign error term to account for map reading error
   reference 'the target coordinates sent to the tubes

   x1 = x + (triag(-100, 0, 100))
   y1 = y + (triag(-100, 0, 100))

   'compute gun-target range

   range = ((x1 - (-500)) ^ 2 + (y1 - (500)) ^ 2) ^ .5

   'compute deflection angle

   k = ATN((500 - y1) / (x1 + 500))
   xmax = x1: xmin = x1: ymax = y1: ymin = y1

   'loop to make round launch sounds
   FOR j = 1 TO 6
     SOUND 250, 2.5
     FOR i = 1 TO RND * 20
        PLAY "l64 n0"
     NEXT i
   NEXT j
```

```
'draw gun-target line (to grid sent to tubes)

LINE (x1, y1)-(-500, 500), 4

'loop to compute impact point of each round

FOR i = 1 TO 6

'compute horizontal and vertical impact point

  x2 = x1 + triag(-.08, 0, .08) * range * SIN(k + 1.571) +
&   triag(-.02, 0, .02) * range * SIN(k)
  y2 = y1 + triag(-.02, 0, .02) * range * COS(k) +
triag(-.08,
&   0, .08) * range * COS(k + 1.571)

'compute the mix and min horizontal and vertical impact
points

  IF x2 > xmax THEN
     xmax = x2
  END IF
  IF x2 < xmin THEN
    xmin = x2
  END IF
  IF y2 > ymax THEN
    ymax = y2
  END IF
  IF y2 < ymin THEN
    ymin = y2
  END IF

'draw explosion and make sound

  PLAY "MBOOL32EFGEFDC"
  Radius = 20
  FOR c# = 0 TO Radius STEP .5
    CIRCLE (x2, y2), c#, 4
  NEXT c#
  FOR j = 1 TO 20
    PLAY "l64 n0"
  NEXT j
NEXT i

'draw casualty box

LINE(xmax + 13.5, ymax + 13.5)-(xmin - 13.5, ymin -
13.5),0,B

'loop to determine if any soldiers were in casualty box

FOR i = 1 TO 12
  IF xmin - 13.5 > soldat(i, 3) OR soldat(i, 3) > xmax +
13.5  &    THEN GOTO 211
  IF ymin - 13.5 > soldat(i, 4) OR soldat(i, 4) > ymax +
13.5
&    THEN GOTO 211
  LOCATE 2, 1: PRINT "Hit on soldier "; i

'updatehit record for mortars for kill card

  wpn = 7
  tgtrec(wpn, 1) = tgtrec(wpn, 1) + 1
  IF tgtrec(wpn, 2) < range THEN tgtrec(wpn, 2) = range
  IF tgtrec(wpn, 4) = 0 THEN tgtrec(wpn, 4) = range
  IF tgtrec(wpn, 4) > range THEN tgtrec(wpn, 4) = range
  tgtrec(wpn, 3) = tgtrec(wpn, 3) + range

'determine results of impact

  CALL impact(i)
  GOTO 213

'determine if soldiers were in suppression box

  211 IF xmin - 100 > soldat(i, 3) OR soldat(i, 3) > xmax +
&  100 THEN GOTO 212
  IF ymin - 100 > soldat(i, 4) OR soldat(i, 4) > ymax + 100
&   THEN GOTO 212
  LOCATE 2, 1: PRINT USING " Soldier ## suppressed"; i
  soldat(i, 10) = 3
  soldat(i, 9) = 0
  GOTO 213

'if no hits were made display message

  212 LOCATE 2, 1: PRINT "No hits"
213 NEXT i

'loop to delay refresh and allow user to view screen
    FOR j = 1 TO 100
    PLAY "164 n0"
    NEXT j
    CALL refresh
END SUB


SUB init
'**********************************************************
'INIT is the subprogram that initializes all data arrays.
'**********************************************************

CLS
SHARED l
SHARED bwire, rwire, rbrch, bbrch
SHARED bluecount, redcount, activeblue, activered
SHARED nexttime
nexttime = 5000
CALL frame(10, 70, 4, 7, 15, 9)
LOCATE 5, 26: PRINT "STANDBY WHILE DATA FILES LOAD"
OPEN "mapl.exp" FOR INPUT AS #1

'read terrain data file

FOR i = 1 TO 50
   FOR j = 1 TO 50
      INPUT #1, mapl(i, j, 1), mapl(i, j, 2), mapl(i, j, 3)
   NEXT j
NEXT i
CLOSE #1

'read soldier attribute file

OPEN "joe.exp" FOR INPUT AS #1
FOR i = 1 TO 12
  FOR j = 1 TO 15
    INPUT #1, soldat(i, j)
  NEXT j
  soldat(i, 16) = soldat(i, 3): soldat(i, 17) = soldat(i, 4)
'update number of starting soldiers

  IF soldat(i, 1) > 0 THEN bluecount = bluecount + 1
  IF soldat(i, 1) < 0 THEN redcount = redcount + 1
NEXT i
CLOSE #1

'read obstacle data file

OPEN "obs.exp" FOR INPUT AS #1
i = 0
bwire = 0
rwire = 0
bbrch = 1
rbrch = 1
DO UNTIL EOF(1)
 bwire = 2 '2=wire in place no id, 1=wire inplace id, 0=no
wire
 rwire = 2
 bbrch = 0
 rbrch = 0
 i = i + 1
 l = i
 INPUT #1, lin(i, 1), lin(i, 2), lin(i, 3), lin(i, 4)
 LOOP
 CLOSE #1

'read initial event file
```

```
OPEN "event.exp" FOR INPUT AS #1
FOR i = 1 TO 99
   FOR j = 1 TO 3
     INPUT #1, event(i, j)
     IF EOF(1) THEN GOTO 1
   NEXT j
NEXT i
1 CLOSE #1

'read threshold Pinf values

OPEN "thresh.dat" FOR INPUT AS #1
FOR i = 1 TO 12
  FOR j = 1 TO 12
    INPUT #1, thresh(i, j)
  NEXT j
NEXT i
CLOSE #1

'read Pinf data for targets not in wooded areas

OPEN "infnw.dat" FOR INPUT AS #1
FOR i = 1 TO 10
  FOR j = 1 TO 4
   INPUT #1, nowoods(i, j)
  NEXT j
NEXT i
CLOSE #1

'read Pinf data for targets in wooded areas

OPEN "infw.dat" FOR INPUT AS #1
FOR i = 1 TO 10
  FOR j = 1 TO 4
    INPUT #1, woods(i, j)
  NEXT j
NEXT i
CLOSE #1

'read cycles resolvable by the observer data

OPEN "cor.dat" FOR INPUT AS #1
FOR i = 1 TO 10
  FOR j = 1 TO 4
    INPUT #1, corl(i, j)
  NEXT j
NEXT i
 CLOSE #1

'read M16 Phit data
'read AK74 Phit data
'read SAW Phit data
```

```
OPEN "p1hit.exp" FOR INPUT AS #1
OPEN "p2hit.exp" FOR INPUT AS #2
OPEN "p3hit.exp" FOR INPUT AS #3
FOR i = 1 TO 8
  FOR j = 1 TO 3
    INPUT #1, p1(i, j)
    INPUT #2, p2(i, j)
    INPUT #3, p3(i, j)
  NEXT j
NEXT i
CLOSE #1: CLOSE #2: CLOSE #3

'read user Phit data

OPEN "p4hit.exp" FOR INPUT AS #1
FOR i = 1 TO 8
  FOR j = 1 TO 3
    IF EOF(1) THEN GOTO 2
    INPUT #1, p5(i, j)
    IF EOF(1) THEN GOTO 2
  NEXT j
NEXT i
2 CLOSE #1

'read user input Phit data

OPEN "p5hit.exp" FOR INPUT AS #1
FOR i = 1 TO 8
  FOR j = 1 TO 3
    IF EOF(1) THEN GOTO 3
    INPUT #1, p6(i, j)
    IF EOF(1) THEN GOTO 3
  NEXT j
NEXT i
3 CLOSE #1

'call the subprogram that allows the user to alter the
'terminating conditions

CALL adjust
END SUB



SUB los (obs, tgt, x, y, x2, y2, r)
'*************************************************************
'LOS is the subprogram that determines line of sight from
'the target to the observer.  LOS is a factor of terrain
'cell elevation, target posture, observer posture, and the
'visibility of the terrain cell (this determines if the
'terrain cell contains wooded areas).  LOS is called by the
'ACQUIRE and the SHOOT subprograms.
'*************************************************************
```

```
SHARED los1

'convert the range (r) to meters

r = r * 1000

'adjust the observer height and target height based on
posture

hexx = INT(x / 20 + 1): hexy = INT(y / 20 + 1)
oht = 1
IF soldat(obs, 10) > 2 THEN oht = .25
IF soldat(obs, 10) = 2 THEN oht = .5
tht = 1
IF soldat(tgt, 10) > 2 THEN tht = .25
IF soldat(tgt, 10) = 2 THEN tht = .5

'compute observer and target elevation

z1 = map1(hexx, hexy, 3) + 1.8 * oht
z2 = map1(INT(x2 / 20 + 1), INT(y2 / 20 + 1), 3) + 1.8 * tht

'compute slope of observer-target line

slope = (z2 - z1) / r

'assign flag w1 = 1 if observer is in a wooded area

IF map1(hexx, hexy, 2) < 1 THEN
    w = 1
 ELSE
    w = 0
END IF

'determine angle of observer-target line

k = ATN(ABS((y2 - y) / (x2 - x)))
IF (y2 > y) AND (x2 < x) THEN k = k + 1.571
IF (y2 < y) AND (x2 < x) THEN k = k - 3.141
IF (y2 < y) AND (x2 > x) THEN k = -k

'loop to check los in every intervening terrain cell

FOR i = 1 TO r
   xn = x + i * COS(k): yn = y + i * SIN(k)

'if already checked elevation of terrain cell try next cell

   IF INT(xn / 20 + 1) = hexx AND INT(yn / 20 + 1) = hexy
&   THEN  GOTO 165

'else compute elevation of present cell
```

```
   hexx = INT(xn / 20 + 1): hexy = INT(yn / 20 + 1)

'check if still in wooded area and adjust elevation
'accordingly

   IF w = 1 AND mapl(hexx, hexy, 2) = 1 THEN w = 0

   IF w = 0 AND mapl(hexx, hexy, 2) < 1 THEN
      znew = mapl(hexx, hexy, 3) + 10 / mapl(hexx, hexy, 2)
   ELSE
      znew = mapl(hexx, hexy, 3)
   END IF

'if elevation of cell does not block los continue to next
cell

   IF znew <= zl + slope * i THEN GOTO 165

'else los is blocked

   losl = 0
   GOTO 169

165 NEXT i
'if checked all cell and los is not blocked, then los exists

losl = 1
169 END SUB


SUB map
'*****************************************************************
'MAP is the subprogram that draws the screen map
'*****************************************************************

SHARED l
SHARED rwire, bwire
SHARED nexttime
CLS
SCREEN 9, , 0, 0
WIDTH 80, 43
WINDOW (0, 0)-(1000, 1000)

'paints the screen the white used to represent clear areas

PAINT (500, 500), 15

'draw the icons for the soldiers and captures them in arrays

x = 100
y = 100
```

```
'draw blue icon first

'draw head
LINE (x - 1, y + 8)-(x + 1, y + 2), 9, BF
LINE (x - 1.5, y + 5)-(x + 1.5, y + 4), 9, BF
'body
LINE (x - 6, y + 2)-(x + 6, y), 9, BF
LINE (x - 3, y + 1)-(x + 4, y - 6), 9, BF
'legs
LINE (x - 1, y - 5)-(x - 3, y - 14), 9, BF
LINE (x + 2!, y - 5)-(x + 4, y - 14), 9, BF
'arms
LINE (x - 6, y + 2)-(x - 7, y - 6), 9, BF
LINE (x + 6, y - 2)-(x + 8, y - 6), 9, BF
GET (93, 108)-(108, 86), barray

'draw red soldier
x = 100: y = 100
LINE (x - 1, y + 8)-(x + 1, y + 2), 4, BF
LINE (x - 1.5, y + 5)-(x + 1.5, y + 4), 4, BF
LINE (x - 6, y + 2)-(x + 6, y), 4, BF
LINE (x - 3, y + 1)-(x + 4, y - 6), 4, BF
LINE (x - 1, y - 5)-(x - 3, y - 14), 4, BF
LINE (x + 2!, y - 5)-(x + 4, y - 14), 4, BF

LINE (x - 6, y + 2)-(x - 7, y - 6), 4, BF
LINE (x + 6, y - 2)-(x + 8, y - 6), 4, BF
GET (93, 108)-(108, 86), rarray

'draw dead icon
x = 100: y = 100
LINE (x - 1, y + 8)-(x + 1, y + 2), 3, BF
LINE (x - 1.5, y + 5)-(x + 1.5, y + 4), 3, BF
LINE (x - 6, y + 2)-(x + 6, y), 3, BF
LINE (x - 3, y + 1)-(x + 4, y - 6), 3, BF
LINE (x - 1, y - 5)-(x - 3, y - 14), 3, BF
LINE (x + 2!, y - 5)-(x + 4, y - 14), 3, BF
LINE (x - 6, y + 2)-(x - 7, y - 6), 3, BF
LINE (x + 6, y - 2)-(x + 8, y - 6), 3, BF
GET (93, 108)-(108, 86), darray

'draw wooded areas

LINE (40, 0)-(0, 380), 2, BF
LINE (40, 0)-(160, 360), 2, BF
LINE (160, 0)-(200, 320), 2, BF
LINE (200, 0)-(220, 300), 2, BF
LINE (220, 0)-(280, 280), 2, BF
LINE (280, 0)-(283, 280), 9, BF
LINE (220, 280)-(283, 283), 9, BF
LINE (220, 280)-(223, 300), 9, BF
LINE (200, 300)-(220, 303), 9, BF
LINE (0, 380)-(40, 383), 9, BF
LINE (40, 360)-(160, 365), 9, BF
LINE (40, 383)-(43, 360), 9, BF
LINE (160, 360)-(163, 320), 9, BF
LINE (160, 320)-(200, 323), 9, BF
LINE (200, 323)-(203, 300), 9, BF

'draw red and black roads

FOR i = 1 TO 25
LINE ((i - 1) * 20, 500 - 2 * (i - 1))-(i * 20, 500 - 3 - 2
* (i - 1)), 4, BF
NEXT i
FOR i = 25 TO 50
LINE ((i - 1) * 20, 449 + 3 * (i - 25))-(i * 20, 449 - 3 + 3
* (i - 25)), 4, BF
NEXT i
LINE (500, 0)-(505, 1000), 0, B
FOR i = 0 TO 1000 STEP 40
LINE (501, i)-(504, i + 20), 4, BF
NEXT i
LINE (480, 455)-(480, 465), 0
LINE (160, 840)-(240, 880), 2, BF

FOR i = 1 TO 50
LINE (480 - (i - 1) * 15, 465 + (i - 1) * 20)-(480 - i * 15,
465 + i * 20), 0
NEXT i
LINE (330, 665)-(250, 665), 0
FOR i = 1 TO 25
LINE (250 - (i - 1) * 20, 665 + (i - 1) * 8)-(250 - i * 20,
665 + i * 8), 0
NEXT i
FOR i = 1 TO 25
LINE (750 + (i - 1) * 10, 0 + (i - 1) * 15)-(750 + i * 10, 0
+ i * 15), 0
NEXT i
FOR i = 1 TO 25
LINE (500 + (i - 1) * 18, 750 - (i - 1) * 9)-(500 + i * 18,
750 - i * 9), 0
NEXT i

'draw contour lines

CIRCLE (150, 880), 50, 6, , , .45
CIRCLE (150, 900), 200, 6, , , .5
CIRCLE (150, 900), 600, 6, , , .2
CIRCLE (1000, 800), 1100, 6, , , .3
LINE (1000, 840)-(900, 720), 2, BF
CIRCLE (1000, 800), 70, 6, , , .2
CIRCLE (1000, 800), 120, 6, , , .25
CIRCLE (50, 50), 100, 6, , , .5
CIRCLE (50, 50), 200, 6, , , .3
```

```
'draw obstacle, if it exists

IF l = 0 THEN GOTO 20
FOR i = 1 TO l
LINE (lin(i, 1), lin(i, 2))-(lin(i, 3), lin(i, 4)), 0
m = (lin(i, 4) - lin(i, 2)) / (lin(i, 3) - lin(i, 1))
B = lin(i, 2) - m * lin(i, 1)
FOR j = 0 TO (lin(i, 3) - lin(i, 1)) STEP 20
x = lin(i, 1) + j
y = lin(i, 2) + m * j
LINE (x - 4, y + 4)-(x + 4, y - 4), 0
LINE (x + 4, y + 4)-(x - 4, y - 4), 0
NEXT j
NEXT i

'draw magenta grid lines
20 FOR i = 200 TO 800 STEP 200
LINE (i, 0)-(i, 1000), 13
LINE (0, i)-(1000, i), 13
NEXT i

'copy screen to alternate screen
PCOPY 0, 1

'put icons on map if the entire icon will fit on the screen

FOR i = 1 TO 12
   IF soldat(i, 3) < 1 OR soldat(i, 3) > 984 THEN GOTO 21
   IF soldat(i, 4) < 1 OR soldat(i, 4) > 979 THEN GOTO 21
   IF soldat(i, 15) < 1 THEN
     PUT (soldat(i, 3), soldat(i, 4)), darray, PSET
     GOTO 21
   END IF
   IF soldat(i, 1) = 1 THEN
     PUT (soldat(i, 3), soldat(i, 4)), barray, PSET
     IF soldat(i, 2) = 1 THEN DRAW "c5 u8 r10 d8 l8"
   ELSE
     PUT (soldat(i, 3), soldat(i, 4)), rarray, PSET
     IF soldat(i, 2) = 1 THEN DRAW "c5 u8 r10 d8 l8"
   END IF
21 NEXT i
nexttime = 5000
END SUB


SUB move (ind, tnow)
'***************************************************************
'MOVE moves all icons.  It updates the soldier location,
'moves the icon, checks the new location for obstacles, and
'schedules the next move.  Moves are scheduled by this
'subprogram, the SHOOT subprogram, and the ENDBREACH
'subprogram.
'***************************************************************
```

```
SHARED nexttime, bmovetime, rmovetime, rbrch, bbrch

'if the soldier is dead or nonmoving, get the next scheduled
'event and schedule another move

IF soldat(ind, 9) = 0 OR soldat(ind, 15) = 0 THEN
   schedule (4,ind,20)
   nexttime = tnow + 100
   GOTO 29
END IF

'if the soldier is BLUE continue

IF soldat(ind, 1) = 1 THEN

'if a breach is in progress, go to the next event

   IF bbrch > 1 THEN
       nexttime = tnow + 100
       GOTO 29
   END IF


'if the icon i s on the map screen erase it

   x = soldat(ind, 16)
   y = soldat(ind, 17)
   IF x > 986 OR x < 0 THEN GOTO 28
   IF y > 979 OR y < 0 THEN GOTO 28
   SCREEN , , 1, 0
   GET (x, y)-(x + 16, y + 24), array2
   SCREEN , , 0, 0
   PUT (x, y), array2, PSET

'compute the new soldier location

   28 soldat(ind, 3)=soldat(ind,3) + 20 * COS(soldat(ind,8))
   soldat(ind,4) = soldat(ind, 4) +20*SIN(soldat(ind, 8))
   x = soldat(ind, 3): y = soldat(ind, 4)

'if the new location will accept an icon, place it on the
'screen

   IF x > 980 OR x < 0 THEN GOTO 29
   IF y > 979 OR y < 0 THEN GOTO 29
   PUT (x, y), barray, PSET

'check the new location for obstacles

   soldat(ind, 16) = x: soldat(ind, 17) = y
   hexx = INT(x / 20 + 1)
   hexy = INT(y / 20 + 1)
   IF mapl(hexx, hexy, 1) < .5 AND bbrch < 1 THEN
```

```
    CALL breach(soldat(ind, 1), tnow)
    nexttime = tnow + 500
    GOTO 29
   END IF


 'if the soldier is a squad leader, draw a box around the
 icon
 'and compute the new movetime

   IF soldat(ind, 2) = 1 THEN
       DRAW "c5 u8 r10 d8 l10"
       hexx = INT(soldat(ind, 3) / 20 + 1)
       hexy = INT(soldat(ind, 4) / 20 + 1)
       IF mapl(hexx, hexy, 1) < .2 THEN
          bmovetime = 10 + 10 * RND / soldat(ind, 10)
        ELSE
          bmovetime = 10 + 10 * RND / (mapl(hexx, hexy, 1) *
 &           soldat(ind, 10))
       END IF
   END IF

 'schedule the next move

 27 CALL schedule(4, ind, tnow + bmovetime)
   GOTO 29  'return

 END IF

 'if the soldier is RED then

 'if RED is breaching, cancel move

 IF rbrch > 1 THEN
   nexttime = tnow + 100
   GOTO 29
 END IF

 'if the soldier's icon is on the screen, erase it

 x = soldat(ind, 16): y = soldat(ind, 17)
 IF x > 986 OR x < 0 THEN GOTO 26
 IF y > 979 OR y < 0 THEN GOTO 26
 SCREEN , , 1, 0
 GET (x, y)-(x + 16, y + 24), array2
 SCREEN , , 0, 0
 PUT (x, y), array2, PSET

 'compute new soldier location

 26 soldat(ind,3) = soldat(ind, 3) + 20 * COS(soldat(ind, 8))
 soldat(ind, 4) = soldat(ind, 4) + 20 * SIN(soldat(ind, 8))
 x = soldat(ind, 3): y = soldat(ind, 4)
```

```
'if the new location is on the map, display it

IF x > 980 OR x < 0 THEN GOTO 29
IF y > 979 OR y < 0 THEN GOTO 29
PUT (x, y), rarray, PSET
soldat(ind, 16) = x: soldat(ind, 17) = y
    hexx = INT(soldat(ind, 3) / 20 + 1)
    hexy = INT(soldat(ind, 3) / 20 + 1)

'check new location for obstacles

    IF mapl(hexx, hexy, 1) = .1 AND rbrch < 1 THEN
    CALL breach(soldat(ind, 1), tnow)
    nexttime = tnow + 500
    GOTO 29
    END IF

'if the soldier is the squad leader, draw a box around the
'icon and compute the next movetime

25  IF soldat(ind, 2) = 1 THEN
        DRAW "c5 u8 r10 d8 l10"
        hexx = INT(soldat(ind, 3) / 20 + 1)
        hexy = INT(soldat(ind, 3) / 20 + 1)
        IF mapl(hexx, hexy, 1) < .2 THEN
           rmovetime = 10 + 10 * RND / soldat(ind, 10)
         ELSE
           rmovetime = 10 + 10 * RND / (mapl(hexx, hexy, 1)
* &                soldat(ind, 10))
        END IF
    END IF

'schedule the next move
  CALL schedule(4, ind, tnow + rmovetime)
29 END SUB


SUB react (tgt, time)
'*****************************************************************
'REACT is the subprogram that determines the reaction of a
'soldier to being shoot at and missed.  The subprogram is
'scheduled by the SHOOT and INDIRECT subprograms.
'*****************************************************************

SHARED nexttime

'pick is the random number used for the Bernoulli trial

pick = RND

'if the soldier is standing then
```

```
 IF soldat(tgt, 10) = 1 THEN
   IF pick <= .5 THEN
      soldat(tgt, 10) = 2
      GOTO 191
   END IF
   IF pick <= .7 THEN
      soldat(tgt, 10) = 3
      GOTO 191
   END IF
   IF pick <= .8 THEN
      soldat(tgt, 10) = 3
      soldat(tgt, 9) = 0
      GOTO 191
   END IF
 GOTO 191
 END IF

 'if the soldier is crouching then

 IF soldat(tgt, 10) = 2 THEN
  IF pick <= .4 THEN
     soldat(tgt, 10) = 3
     GOTO 191
  END IF
  IF pick <= .5 THEN
     soldat(tgt, 10) = 3
     soldat(tgt, 9) = 0
     GOTO 191
  END IF
  GOTO 191
 END IF

 'if the soldier is prone

 IF soldat(tgt, 10) = 3 THEN
   IF pick <= .5 THEN
      soldat(tgt, 9) = 1
    ELSE
      soldat(tgt, 9) = 0
   END IF
 END IF

 'update the history attribute to reflect current posture

 191 soldat(tgt, 7) = soldat(tgt, 10)

 'if the soldier is a squad leader then ensure the rest of
 'his squad adopts the new posture

 IF soldat(tgt, 2) = 1 THEN
    side = soldat(tgt, 1)
    FOR i = 1 TO 12
      IF soldat(i, 1) = side THEN
```

```
        soldat(i, 9) = soldat(tgt, 9)
        soldat(i, 10) = soldat(tgt, 10)
        soldat(i, 7) = soldat(tgt, 10)
      END IF
    NEXT i
  END IF
  nexttime = time + 100
  END SUB



  SUB refresh
  '*************************************************************
  'REFRESH is a program that refreshes the screen.  It copies
  'the map screen off the hidden screen and then places the
  'icons on it.  REFRESH is scheduled by the BREACH
  subprogram.
  '*************************************************************

  'copy the map screen

  PCOPY 1, 0

  'place icons on the map

  FOR i = 1 TO 12
    IF soldat(i, 3) < 1 OR soldat(i, 3) > 984 THEN GOTO 160
    IF soldat(i, 4) < 1 OR soldat(i, 4) > 979 THEN GOTO 160
    IF soldat(i, 15) = 0 THEN
      PUT (soldat(i, 3), soldat(i, 4)), darray, PSET
      GOTO 160
    END IF
    IF soldat(i, 1) > 0 THEN
      PUT (soldat(i, 3), soldat(i, 4)), barray, PSET
      IF soldat(i, 2) = 1 THEN DRAW "c5 u8 r10 d8 18"
    ELSE
      PUT (soldat(i, 3), soldat(i, 4)), rarray, PSET
      IF soldat(i, 2) = 1 THEN DRAW "c5 u8 r10 d8 18"
    END IF
  160 soldat(i, 16) = soldat(i, 3)
  soldat(i, 17) = soldat(i, 4)
  NEXT i
  nexttime = 5000
  END SUB



  SUB schedule (act, ind, T)
  '*************************************************************
  'SCHEDULE adds events to the event calendar.  It is called
  'by all subprograms that schedule events.
  '*************************************************************
```

```
SHARED nexttime

'search the event calendar for an empty matrix row

FOR i = 1 TO 90
  IF event(i, 1) > 0 THEN GOTO 31
  event(i, 1) = act
  event(i, 2) = ind
  event(i, 3) = T
  GOTO 32
31 NEXT i
32 nexttime = T
END SUB



SUB selct (obs, time)
'**********************************************************
'SELECT is the subprogram that allows soldiers to select a
'target to engage from the targets on their target list.  It
'is scheduled by ACQUIRE whenever one or more targets has
'been detected.
'**********************************************************

SHARED nexttime
IF soldat(obs, 15) = 0 THEN
 nexttime = 5000
 GOTO 103
 END IF

'inititialize the count of number of targets

 count = 0

'loop to count number of potential targets and sum Pdet
'values

 FOR i = 1 TO 12
   IF ptgt(obs, i) <= 0 THEN GOTO 101
   total = ptgt(obs, i) + total
   count = count + 1
101 NEXT i

'if observer is a squad leader schedule a formation change

IF soldat(obs, 2) = 1 THEN CALL schedule(7, obs, time + 20)
pick = RND
T = 0

'loop to pick a target

FOR j = 1 TO 12
```

```
  IF ptgt(obs, j) <= 0 THEN GOTO 102

'normalize all pdet values

  target = ptgt(obs, j) / total
  T = T + target
  IF pick < T THEN

' if the soldier is a BLUE squad leader and has detected
more 'than one target, schedule an indirect fire event

  IF soldat(obs, 2) = 1 AND soldat(obs, 1) > 0 THEN
    IF count > 1 THEN
      soldat(obs, 14) = j
      x1 = soldat(obs, 3)
      y1 = soldat(obs, 4)
      x2 = soldat(j, 3): y2 = soldat(j, 4)
      range = ((x1 - x2) ^ 2 + (y1 - y2) ^ 2) ^ .5

'if range to target is less than 100m, do not request fire

      IF range > 100 THEN CALL schedule(9, j, time + 20)
    END IF
  END IF

'schedule a direct fire engagement

  CALL schedule(3, obs, time + 7 + 6 * (RND - .5))

'change firer to a nonmove status

  soldat(obs, 9) = 0

'assign target to firer

  soldat(obs, 14) = j

'record firer's previous posture

  soldat(obs, 7) = soldat(obs, 10)

'reduce firer's posture one level

  IF soldat(obs, 10) = 3 > 1 THEN soldat(obs,10) =
&       soldat(obs,10) - 1

'end select

  GOTO 103
 END IF
102 NEXT j
nexttime = time + 100
103 END SUB
```

```
SUB shoot (obs, time)
'************************************************************
'SHOOT is the subprogram that processes all direct fire
'events.  The subprogram is scheduled by the SELECT target
'subprogram.
'************************************************************

SHARED nexttime
SHARED losl
IF soldat(obs, 15) = 0 THEN
 nexttime = 5000
 GOTO 119
END IF

'B is the variable that represents the number of rounds per
'burst.  It is used to control sound.

B = 3

'ID target, observer location, target location and range

tgt = soldat(obs, 14)
xl = soldat(obs, 3): yl = soldat(obs, 4)
x2 = soldat(tgt, 3): y2 = soldat(tgt, 4)
range = (((xl - x2) ^ 2 + (yl - y2) ^ 2) ^ .5)

'check line of sight to target

CALL los(obs, tgt, xl, yl, x2, y2, range / 1000)

'if no los, then that target is removed from the potential
'target list and must be reacquired.

IF (losl < 1) THEN
   ptgt(obs, tgt) = 0
   'schedule another SEARCH
   CALL schedule(1, obs, time + 10 + 10 * (RND - .5))
   GOTO 119
END IF

'go to line based on weapon type

ON soldat(obs, 11) GOTO 111, 112, 113, 114, 115, 116

'M16 Phit

111 IF soldat(obs, 12) < 3 GOTO 118  'check ammo
'assign Phit
phit = pl(INT((range + 50) / 100), soldat(tgt, 10))
decrement ammo count
soldat(obs, 12) = soldat(obs, 12) - 3
GOTO 117
```

```
'AK74 Phit

112 IF soldat(obs, 12) < 3 GOTO 118  'check ammo
phit = p2(INT((range + 50) / 100), soldat(tgt, 10))
soldat(obs, 12) = soldat(obs, 12) - 3
GOTO 117

'SAW Phit

113 B = 6
IF soldat(obs, 12) < 6 GOTO 118  'check ammo
IF range < 350 THEN
  phit = p3(INT((range + 50) / 100), soldat(tgt, 10))
soldat(obs, 12) = soldat(obs, 12) - 6
GOTO 117
END IF
 soldat(obs, 12) = soldat(obs, 12) - 6
 IF 750 > range > 349 THEN
  phit = p3(4, soldat(tgt, 10))
  GOTO 117
 ELSEIF 750 < range < 900 THEN
  phit = p3(5, soldat(tgt, 10))

  GOTO 117
   ELSE
    phit = p3(6, soldat(tgt, 10))
    GOTO 117
END IF

'M203 Phit

114 IF range > 300 THEN  'make weapon choice based on range
       IF soldat(obs, 12) < 3 GOTO 118
       B = 3
       phit = p1(INT((range + 50) / 100), soldat(tgt, 10))
       soldat(obs, 12) = soldat(obs, 12) - 3
       GOTO 117
END IF

'If range < 300, use M203

B = 1
soldat(obs, 5) = soldat(obs, 5) - 1
IF soldat(obs, 5) < 1 THEN
  soldat(obs, 11) = 1
END IF

'assign impact point

x = x2 + triag(-.032, 0, .032) * range
y = y2 + triag(-.096, 0, .096) * range

'compute Phit using Carlton's Function
```

```
phit = EXP(-((x - x2) ^ 2 + (y - y2) ^ 2) / 25)
GOTO 117

'User input weapon Phit

115 IF soldat(obs, 12) < 3 GOTO 118
phit = p5(INT((range + 50) / 100), soldat(tgt, 10))
 soldat(obs, 12) = soldat(obs, 12) - 3
GOTO 117

'user input weapon Phit

116 IF soldat(obs, 12) < 3 GOTO 118
phit = p6(INT((range + 50) / 100), soldat(tgt, 10))
 soldat(obs, 12) = soldat(obs, 12) - 3

'assign firing time to soldier attribute 6, this increases
'firers' signature

117 soldat(obs, 6) = time

'compute angle of observer-target line

k = ATN(ABS((y2 - y1) / (x2 - x1)))
IF (y2 > y1) AND (x2 < x1) THEN k = 3.141 - k
IF (y2 < y1) AND (x2 < x1) THEN k = k - 3.141
IF (y2 < y1) AND (x2 > x1) THEN k = -k
IF (y1 = y2) AND (x2 < x1) THEN k = 3.141
x = x1: y = y1 + 10

'draw line of fire

FOR i = 1 TO range STEP 5
LINE (x, y)-(x1 + i * COS(k), y1 + 10 + i * SIN(k)), 4
x = x1 + i * COS(k): y = y1 + 10 + i * SIN(k)
NEXT i

'if the grenade launcher is used, make the explosion

IF B = 1 THEN
  CALL explode(x, y, 10)
  CALL refresh
  GOTO 120
END IF

'if direct fire is used, make a sound to represent rounds
'firing

PLAY "t80"
FOR i = 1 TO B
SOUND 250, 2.5
PLAY "l64 n0"
NEXT i
```

```
'erase the red line

x = xl: y = yl + 10
FOR i = 1 TO range STEP 5
LINE (x, y)-(xl + i * COS(k), yl + 10 + i * SIN(k)), 15
x = xl + i * COS(k): y = yl + 10 + i * SIN(k)
NEXT i
LINE (x, y)-(xl + range * COS(k), yl + 10 + range *
SIN(k)),15

'Bernoulli trial to determine outcome of engagement

120 IF RND > phit THEN
     LOCATE 2, 1: PRINT USING "Miss   P(hit) =   .###"; phit
     CALL schedule(6, tgt, time + 3) 'schedule react to fire
    ELSE
     LOCATE 2, 1: PRINT USING "Hit    P(hit) =   .###"; phit

'assign weapon type for recording hit data for kill card

     wpn = soldat(obs, 11)
     IF soldat(obs, 1) < 1 AND wpn = 3 THEN wpn = 8
     IF wpn = 4 AND B > 1 THEN wpn = 1
     tgtrec(wpn, 1) = tgtrec(wpn, 1) + 1
     IF tgtrec(wpn, 2) < range THEN tgtrec(wpn, 2) = range
     IF tgtrec(wpn, 4) < 1 THEN tgtrec(wpn, 4) = range
     IF tgtrec(wpn, 4) > range THEN tgtrec(wpn, 4) = range
     tgtrec(wpn, 3) = tgtrec(wpn, 3) + range

'call impact to see results of hit

     CALL impact(tgt)
   END IF

'random number draw to see whether to reengage the target

 IF RND > .7 THEN   'schedule another engagement
       CALL schedule(3, obs, time + 5)
       GOTO 119
  ELSE   'schedule a search
      soldat(obs, 9) = 1
      x = time + 5 * (RND - .3) * 5
      IF range > 50 THEN
        soldat(obs, 9) = 1
        soldat(obs, 10) = soldat(obs, 7)
      ELSE  'don't move if range < 50m
        soldat(obs, 9) = 0
      END IF
      CALL schedule(1, obs, x + 3)
      GOTO 119
  END IF
```

```
'adjusts magazine and ammo count to reflect reloading

118 soldat(obs, 13) = soldat(obs, 13) - 1
IF soldat(obs, 10) = 1 THEN soldat(obs, 12) = 30
IF soldat(obs, 10) = 2 THEN soldat(obs, 12) = 40
IF soldat(obs, 10) = 3 THEN soldat(obs, 12) = 200
IF soldat(obs, 10) = 5 THEN soldat(obs, 12) = 30
IF soldat(obs, 10) = 6 THEN soldat(obs, 12) = 30

'schedule a search

CALL schedule(3, obs, time + 5 + (RND - .3) * 5)
119 END SUB



FUNCTION triag (a!, d!, B!) STATIC
'*************************************************************
'This function is used to replicate the normal distribution
'*************************************************************
'VARIABLES:  a = lower bound = mode - 2 stddev
'            d = mode
'            B = upper bound = mode + 2 stddev

r = RND
IF (r < (d - a) / (B - a)) THEN
triag = a + SQR((d - a) * (B - a) * r)
ELSE
triag = B - SQR((B - d) * (B - a) * (1 - r))
END IF
END FUNCTION


SUB wire (obs)
'*************************************************************
'WIRE is called from the ACQUIRE subprogram, if a wire
'obstacle has been input the user.  The subprogram first
'determines if the current azimuth of the squad leader
'intersects the obstacle.  If it does, the subprogram checks
'if the squad leader has line of sight to the obstacle.  If
'so, a message reflecting obstacle detection is displayed on
'the screen.  The subprogram is not accessed again by that
'side, unless that side makes a direction change.  In that
'case, the obstacle flag is updated and ACQUIRE again calls
'WIRE.
'*************************************************************

SHARED bwire, rwire, l, bbrch, rbrch
IF soldat(obs, 1) = 0 THEN 129

'1st compute the constants of the squad leader's azimuth and
```

```
'the wire obstacle linear equations

bl = soldat(obs, 3): y = soldat(obs, 4)
x = bl
a = soldat(obs, 8)'  a = slope of line
i = 1

'lin is the array containing the horizontal and vertical
start ;and end points of the obstacle

'm = slope of obstacle line
m2 = (lin(i, 4) - lin(i, 2)) / (lin(i, 3) - lin(i, 1))
b2 = y intercept of obstacle line
b2 = lin(i, 2) - m2 * lin(i, 1)

'loop checks every meter along the obstacle to see if a line
'having the slope (azimuth) of the squad leader) intersects
'the obstacle

FOR j = lin(i, 1) TO lin(i, 3)
   r = ((j - x) ^ 2 + (m2 * j + b2 - y) ^ 2) ^ .5
   IF ABS(x + r * COS(a) - j) > 1 THEN GOTO 127'no intersect
   IF ABS(y + r * SIN(a) - j * m2 - b2) > 1 THEN GOTO 127
   'hexx = horizontal coordinate of terrain cell

   hexx = INT(x / 20 + 1)

'hexy = vertical coordinate of terrain cell

   hexy = INT(y / 20 + 1)

'adjust elevation to reflect current posture of squad leader

   oht = 1
   IF soldat(obs, 10) > 2 THEN oht = .25
   IF soldat(obs, 10) = 2 THEN oht = .5

'compute height of observer and obstacle

   z1 = mapl(hexx, hexy, 3) + 1.8 * oht
   z2 = mapl(INT(j / 20 + 1), INT((m2 * j + b2) / 20 + 1),
&   3)+1.5

IF mapl(hexx, hexy, 2) < 1 THEN
    w = 1
 ELSE
    w = 0
END IF

'slope of observer-obstacle line

   slope = (z2 - z1) / r
```

```
'  k = azimuth

   k = soldat(obs, 8)

'loop to check los to obstacle, same method as the los
   FOR h = 1 TO r
     xn = x + h * COS(k): yn = y + h * SIN(k)
     IF INT(xn / 20 + 1) = hexx AND INT(yn / 20 + 1) = hexy
&    THEN  GOTO 128
     hexx = INT(xn / 20 + 1): hexy = INT(yn / 20 + 1)

'check if still in wooded area and adjust elevation
'accordingly

   IF w = 1 AND mapl(hexx, hexy, 2) = 1 THEN w = 0

   IF w = 0 AND mapl(hexx, hexy, 2) < 1 THEN
      znew = mapl(hexx, hexy, 3) + 10 / mapl(hexx, hexy, 2)
   ELSE
      znew = mapl(hexx, hexy, 3)
   END IF

'if the new elevation is less than the previous cell, no los

     IF znew <= zl + slope * h THEN GOTO 128
     GOTO 129
  128 NEXT h

'if no intervening terrain blocks line of sight display '
'message and update obstacle detection flag

  IF soldat(obs, 1) > 0 THEN
       strng$ = "Blue"
       bwire = 1
  ELSE
       strng$ = "Red "
       rwire = 1
  END IF
  LOCATE 1, 1
  PRINT USING "& soldier detects obstacle at ### ###";
&  strng$; j; m2 * j + b2
  GOTO 129


127 NEXT j
'if the outer loop has cycled all the way through, then the
'current azimuth does not intersect the obstacle and a
'bypass is possible.  Update the obstacle breach status and
'the obstacle detection status

IF soldat(obs, 1) > 0 THEN
  bwire = 2
  bbrch = 1
```

```
 ELSE
  rwire = 2
  rbrch = 1
 END IF
129 END SUB
```

Bibliography

1. Anderson, L. B., J.H. Cushman, A. L. Gropman, V.P. Roske. "A Toxonomy for Warfare Simulation" A Workshop Report. Military Operations Research Society, 1987.

2. Battilega, John A. and Judith K. Grange. The Military Applications of Modeling. Washington: Government Printing Office, 1984.

3. Bonder, Seth. "An Overview of Land Battle Modeling in the US," Proceedings 13th U.S. Army Operations Research Symposium: 73-88, (November 1974)

4. Brewer, Garry D. and Martin Shubik. The War Game: A Critique of Military Problem Solving. Cambridge: Harvard University Press, 1979.

5. Cox, CPT David K. SPARTAN: An Instuctional High Resolution Land Combat Model. MS Thesis, AFIT/ENS/GOR/92M-7. Air Force Institute of Technology, Wright-Patterson AFB OH, March 1992 (No DTIC yet)

6. Department of the Army. Army Model Improvement Program. AR 5-11 (draft). Washington: Government Printing Office, 1990.

7. Department of the Army. US Army TRADOC Analysis Center. CASTFOREM Update: Methodologies. TRAC-WSMR-TD-92-011. Washington: Government Printing Office, April 1992.

8. Department of the Army. US Army TRADOC Analysis Center. JANUS (T) Documentation. Washington: Government Printing Office, June 1986.

9. Dunnigan, James F. The Complete Wargames Handbook. New York: William Morrow and Company, 1980.

10. Engineering Design Handbook, Army Weapon Systems Analysis, Part One, DARCOM-P 706-101. U.S. Army Material Development and Readiness Command, Washington: Government Printing Office, November 1977.

11. Government Accounting Office, Guidelines for Model Evaluation: Exposure Draft. PAD-79-17, Washington: Government Printing Office, January 1979.

12. Government Accounting Office. Models, Data, and War: A Critique of the Foundation for Defense Analysis. Washington: government Printing Office, 1980.

13. Hartman, James K. Lecture Notes in High Resolution Combat Modeling. Unpublished Not s, 1985. Class handout for OPER 775, Land Combat Modeling I. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, July 1991.

14. Joint Analysis Directorate, Organization of the Joint Chiefs of Staff. Catalogue of Wargaming and Military Simulations Modeling. JADAM 207-91. Washington: Government Printing Office, 1991.(AD-A213-970)

15. Krueger, John L. "Pitfalls in Combat Simulations," Military Review LXXII: 20-25. (June 1992).

16. Law, Averill M. and W David Kelton. Simulation Modeling and Analysis. New York: McGraw-Hill Book Company, 1982.

17. Lawrence Livermore National Laboratory's Conflict Simulation Laboratory. The JANUS Algorithms Guide. California: University of California, 1990.

18. Nance, Richard E. and James D. Arthur. "The Methodology Roles in the Realization of a Model Development Environment," Proceedings of the 1988 Winter Simulation Conference. 220-225. New York: IEEE Press, 1988.

19. Pritsker, A. Alan B. Introduction to Simulation and SLAM II. New York: Halsted Press Book, 1986.

20. Rand Corporation. Systems Analysis and Policy Planning: Applications in Defense. Edited by Quade, E. S. and W. J.Boucher. New York: Elsevier, 1968.

21. Ross, John G. "An exclusive AFJI interview with: General Frederick M. Franks, Jr., USA," Armed Forces Journal International. 68-69. (October 1992).

Vita

Captain Edwin H. Harris III was born on 29 June 1961 in Durham, North Carolina. He graduated from Lake Braddock Secondary School in Fairfax, Virginia in 1979 and entered the United States Military Academy in July 1979. He graduated from West Point, with a Bachelor of Science degree in Civil Engineering, in 1983.

Upon graduation, he was commissioned as an Infantry officer. After a series of military schools at Fort Benning, Georgia, CPT Harris was assigned to the 82nd Airborne Division at Fort Bragg, North Carolina. While there, he served as an infantry platoon leader, a rifle company executive officer, and a battalion air operation's officer.

In 1987, after attending the Infantry Officer's Advanced Course, CPT Harris was assigned to a mechanized infantry battalion in the 1st Armored Division in Germany. During his four years in Germany, CPT Harris served as a battalion adjutant, company commander, and brigade plans officer.

In August 1991, after his assignment in Germany, CPT entered the School of Engineering, Air Force Institute of Technology.

Permanent Address: 2007 Stonegate
Denton, TX 76205

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1993	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE SPARTAN II: AN INSTRUCTIONAL HIGH RESOLUTION LAND COMBAT MODEL	5. FUNDING NUMBERS
6. AUTHOR(S) CPT Edwin H. Harris III, USA	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GOR/ENS/93M-09
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) ENS/AFIT Wright-Patterson AFB, Ohio 45433	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unlimited Distribution	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This project improved SPARTAN, a high resolution land combat model demonstrator. SPARTAN was originally developed as a hands-on trainer for land combat modeling students. The new SPARTAN is built to demonstrate the techniques used in the current generation of US Army high resolution models. Like the original, this model is primarily a small scale attrition (both direct and indirect fire) model. The model represents 12 soldiers involved in the following processes: target search, target selection, direct fire engagement, indirect fire engagement, movement, reaction to fire, obstacle breaching, and some elements of command and control. The emphasis on model development was to keep the logic simple, yet accurately portray current modeling techniques as used in JANUS and CASTFOREM. SPARTAN contains numerous features that allow the user to observe, in great detail, how the model represents the various activities of the soldiers. An educational assessment of the model was performed by students and faculty at the Air Force Institute of Technology.

14. SUBJECT TERMS Combat Patrols, Models, High Resolution Weapons Effects	15. NUMBER OF PAGES 261
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18